mi COMPUTER

29

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR



Editorial Delta, S.A.

150 ptas.

mi computer CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona, y comercializado en exclusiva por Distribuidora Olimpia, S.A., Barcelona

Volumen III - Fascículo 29

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Jesús Nebra

Redactores y colaboradores: G. Jefferson, R. Ford, S. Tarditti. A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, Barcelona-8
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador,* se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-85822-94-3 (tomo 3)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 018408
Impreso en España - Printed in Spain - Agosto 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, Madrid-34.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio Blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Ferrenquín a Cruz de Candelaria, 178, Caracas, y todas sus sucursales en el interior del país.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 3371872 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Distribuidora Olimpia (Paseo de Gracia, 88, 5.°, Barcelona-8), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Distribuidora Olimpia, en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

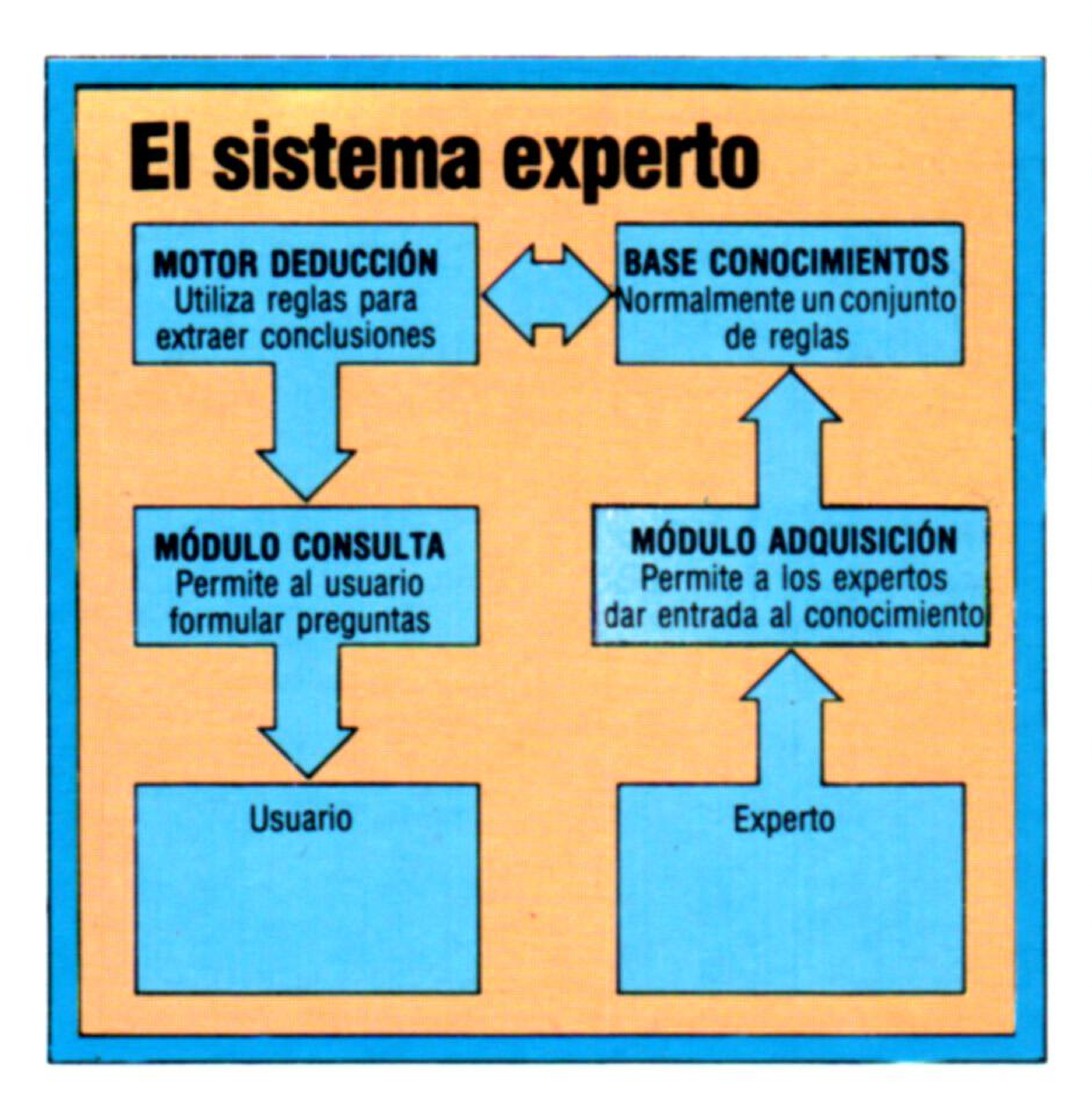
No se efectúan envíos contra reembolso.





El increíble HULK

Los sistemas expertos, en otros tiempos campo de acción exclusivo de los laboratorios de investigación, se encuentran ahora al alcance de los usuarios de ordenadores personales



EL HULK

Archivo principal de datos

Datos de prueba

Conjunto de entrenamiento

Archivo de reglas

LOOK

LEAP

Utiliza las reglas

Prueba las reglas

Pronósticos

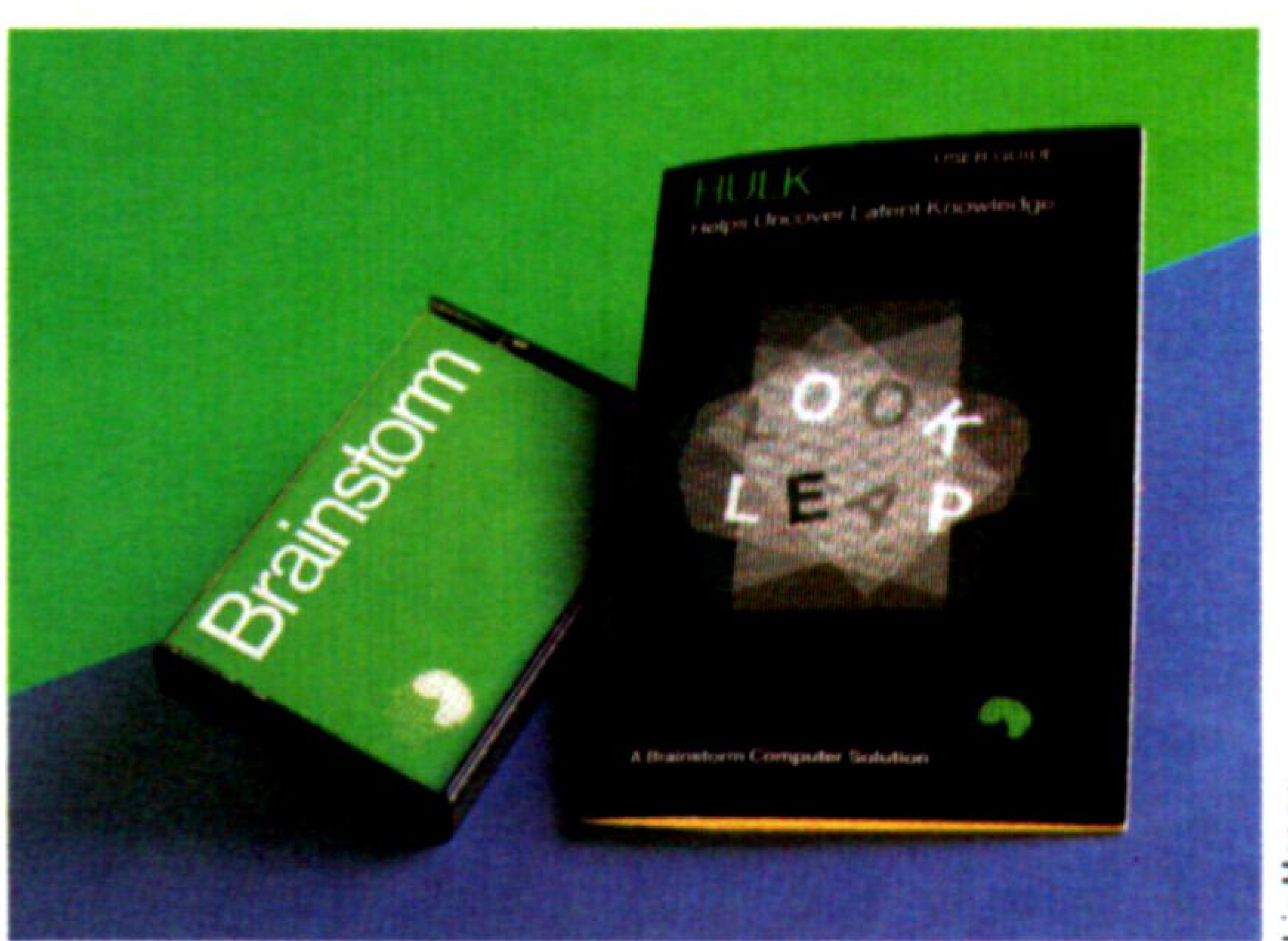
Liz Dixon

La eficacia del HULK
En un sistema experto tradicional, una serie de reglas IF...THEN, retenidas en la base de conocimientos, son aplicadas por el motor de deducción en respuesta a los interrogantes del usuario, que entra en el sistema a través del módulo de consulta. El conocimiento sobre el cual se basan estas reglas se introduce en el sistema experto a través del módulo de adquisición. Con el HULK se construye un conjunto de reglas de decisión a partir de un archivo principal de datos que contiene observaciones a las que da entrada el usuario. Se necesitan dos programas separados: el LOOK, que requiere de un conjunto de entrenamiento de datos para probar las reglas y de un conjunto de prueba de datos para verificar su utilidad, y el LEAP, que emplea el archivo de reglas y los datos de prueba para producir un pronóstico de probabilidades. El HULK se ha descrito como "el sistema experto del pobre"

El término "ingeniería" está experimentando un cambio en su significado. En el siglo XIX, un ingeniero era alguien como Brunel (1806-1859), que transformaba el hierro y el acero en barcos y puentes. En la actualidad han hecho suya esta palabra diversas profesiones de despacho que "diseñan" materias primas como el "conocimiento".

Un ingeniero de conocimiento es alguien que sabe cómo construir un *sistema experto* (SE). Un sistema experto sintetiza el conocimiento organizado acerca de algún campo de la experiencia humana. Consideremos, por ejemplo, el caso del diagnóstico médico: un médico posee un gran acopio de conocimientos basados en los hechos acerca de las enfermedades y de sus signos y síntomas; la experiencia del médico reside en la capacidad de relacionar el estado de un paciente con las descripciones de las condiciones típicas proporcionadas por el libro de texto. Al hacerlo, el médico determina qué síntomas están presentes y coteja su significado con el de los síntomas ausentes y/o la enfermedad ante la cual cree encontrarse. Cuanto mayor sea la capacidad del médico para combinar el conocimiento de los libros con las observaciones reales, tanto más exacta será la técnica de diagnóstico. Los factores limitativos son la capacidad de recordar datos organizados, la capacidad de relacionar casos observados con el patrón de los datos existentes y la capacidad de aplicar este conocimiento en casos en los que los datos son incompletos o no se adaptan muy bien a los casos anteriores. Estos dos primeros factores (la organización y la clasificación de datos) son el punto fuerte de los ordenadores; el último factor es el punto fuerte de los expertos humanos. Si un sistema informático puede sustituir la "percepción" o el "olfato" del experto humano por el análisis estadístico, entonces sus superiores poderes para la organización de datos podrían capacitarlo para superar al experto.

Los sistemas de este tipo son más relevantes allí donde sea necesario el juicio humano por no existir una teoría completa, como en el diagnóstico médico. También se pueden aplicar provechosamente allí donde, aun cuando esté disponible a nivel público el conocimiento requerido, éste sea demasia-



Liz Heaney

Proyecto práctico
Al principio Richard Forsyth concibió el HULK como un plan práctico para sus alumnos del Polytechnic of North London. Si bien escribir el programa sólo le ocupó dos semanas, se necesitaron seis meses para mejorarlo y refinarlo con el fin de hacerlo más accesible al usuario. Es posible que pronto aparezca una versión para el QL

do complicado como para que lo apliquen la mayoría de las personas (p. ej., la legislación acerca de la declaración de la renta o las disposiciones que regulan el derecho a disfrutar de los beneficios de la Seguridad Social). Debido a que para la operación de este tipo de sistemas el conocimiento, frecuentemente expresado como un conjunto de reglas, resulta esencial, también se los conoce como sistemas *basados en reglas* o *basados en el conocimiento*.

Los sistemas expertos han logrado un éxito sorprendente en varios campos. Ya pueden superar en rendimiento a los profesionales humanos experimentados en diagnóstico médico, en prospección de minerales o en muchos otros campos. En virtud de estos notables éxitos, estos sistemas están siendo creados en gran número por los laboratorios de investigación de inteligencia artificial y encontrando aplicaciones en la práctica de la informática general, con importantes consecuencias por la forma en que la gente está construyendo sistemas informáticos de elevado rendimiento. De cara al diseño de software, el enfoque basado en el conocimiento reemplaza la tradición de:

INFORMACION + ALGORITMO = PROGRAMA

por una metodología basada en:

CONOCIMIENTO + DEDUCCION = SISTEMA

que es un cambio evolutivo de consecuencias muy significativas.

De manera que un sistema experto se fundamenta en una *base de conocimientos*. En un SE totalmente desarrollado existen en realidad otros tres componentes básicos: el *motor de deducción*, que forma nuevas reglas para interpretar los datos sobre la base de las reglas existentes y los datos; el *módulo de adquisición de conocimientos*, a través del cual el sistema adquiere conocimientos a partir de su propia experiencia y la de los expertos humanos, y la *interface para el usuario*, que permite que los no expertos tengan la posibilidad de interrogar al sistema y de utilizarlo.

Una base de conocimientos contiene hechos (o aseveraciones) y reglas. Los hechos pueden cambiar rápidamente; por ejemplo, durante el curso de una consulta. Las reglas son la información a más largo plazo acerca de cómo generar nuevos hechos o hipótesis a partir de lo que se conoce actualmente. ¿En qué difiere esto de una base de datos convencional? La diferencia fundamental estriba en que una base de conocimientos es más creativa. Los hechos de una base de datos normalmente son pasivos: o están o no están allí, para ser archivados o recuperados por el usuario en la medida en que éste así lo requiera. Una base de conocimientos, por el contrario, intenta rellenar activamente la información que falta, a la luz de lo que ya "sabe" e independientemente de las exigencias inmediatas de datos.

Las reglas que responden al formato IF...THEN (conocidas como "reglas de producción") son un método válido para expresar el conocimiento de "reglas empíricas". Por ejemplo:

IF (si) el equipo local perdió el último partido jugado en su campo, AND (y) el equipo visitante ganó por dos goles el último partido jugado en su campo
THEN (entonces) la probabilidad de un empate se multiplica por 1 088.

Pronóstico meteorológico

Los archivos de datos del **HULK** los crea el usuario como archivos de programas en BASIC, por lo que resultan fácilmente accesibles para la inspección y edición. El archivo que utilizamos es típico.

Ejecutamos el **HULK** empleando este archivo y se nos solicitó una hipótesis acerca de los datos; la hipótesis fue:

MAÑANA = 1

con lo que queremos significar que nos interesan aquellas muestras en las que la última variable posee el valor 1; en otras palabras, los días en que llovió al día siguiente. Deseamos utilizar el **HULK** para establecer reglas que nos permitan predecir el tiempo que hará mañana a partir del tiempo que hace hoy.
Seguidamente estamos en condiciones de dar entrada a reglas para prueba. Cada regla se aplica al archivo de datos y el éxito que obtiene en cuanto a predecir lluvia se mide y se informa, tanto como regla única como cuando se utiliza en combinación con reglas anteriores. El **HULK** aconseja acerca de si agregar o no la regla al conjunto de reglas, pero la decisión en este sentido queda en manos del usuario.

Rápidamente encontramos tres reglas para pronosticar lluvia para mañana:

1) Si hoy llueve más de 2 mm
AND (y)
2) Si hoy el sol brilla menos de 3,5 horas
AND (y)
3) Si la temperatura máxima es superior en menos de 6 grados a la temperatura mínima de hoy
THEN (entonces)
uno puede tener una certeza de un 83 % al pronosticar lluvia para mañana (siempre que, por supuesto, los datos del tiempo relativos a este mes sean una muestra representativa de los de todo el año).

Aunque éste es un archivo de programas en BASIC, no está pensado para ejecutarse; es simplemente una forma conveniente de almacenar datos.

```
1TIEMPO,30,5
```

La línea 1 describe el archivo: contiene su nombre (TIEMPO), el número de muestras de datos (30) y el número de datos (5) por muestra. El archivo contiene los datos del tiempo para un mes: cada día se describe en lo que se refiere a temperatura mínima y máxima (en centígrados), lluvia diaria (en mm), horas de sol y una variable booleana cuyo valor es 1 si lloverá al día siguiente y 0 si no lloverá.

```
100 TMINIMA
200 TMAXIMA
300 LLUVIA
400 SOL
500 MANANA
```

Desde la línea 100 a la 500 se proporcionan los nombres de variables para los datos de cada muestra

```
1001 D01,54,110,175,32,1
1002 D02,42,125,041,62,1
1003 D03,76,112,077,11,1
1004 D04,27,105,018,43,0
1005 D05,30,120,000,95,0
1006 D06,44,106,000,55,0
1007 D07,48,094,000,51,1
1008 D08,68,092,055,48,1
1009 D09,64,102,048,41,1
     -----  -----
1028 D28,58,154,000,20,1
1029 D29,67,088,064,42,0
1030 D30,45,096,000,68,1
```

Entre las líneas 1001 y 1030 se incluyen los datos, con una muestra diaria por línea del programa, y empezando cada línea con una etiqueta que identifica esa muestra. Los valores del HULK han de ser números enteros, así que todos los datos se han multiplicado por diez para conservar la precisión:
La línea 1001, por ejemplo, en realidad debería decir:
1001, D01,5.4,11.0,17.5,3.2,1 y así sucesivamente para el resto del archivo

MAÑANA = 1

SOLICITANDO REGLA **1**

LA REGLA ES: **LLUVIA > 20**

TABLA DE CONTINGENCIA	ÉXITO	FRACASO
REGLA VERDADERA	12	2
REGLA FALSA	5	11

PROMEDIO DE ÉXITO = **76,3 %**

BITS POR MUESTRA

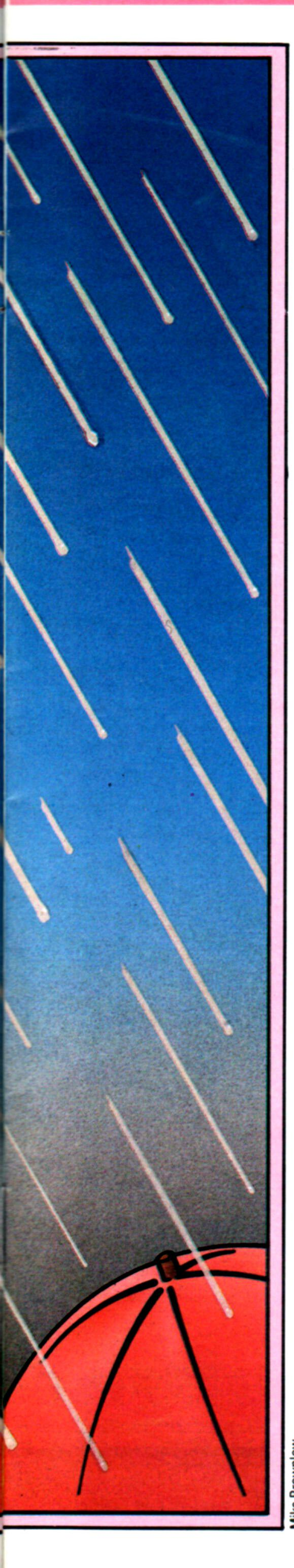
ANTES	**LLUVIA > 20**	**1,00**
DESPUÉS	**LLUVIA > 20**	**0,85**

SE LE ACONSEJA CONSERVAR LA REGLA

LA REGLA SE AÑADE AL CONJUNTO DE REGLAS

Bits por muestra
Ésta es una medida de cuántas de las muestras de datos están contribuyendo al éxito de la regla





Mike Brownlow

Para inferir deducciones existen dos estrategias de alto nivel fundamentales: la "encadenación hacia adelante" y la "encadenación hacia atrás". En términos generales, la encadenación hacia adelante implica examinar los datos por orden para formular hipótesis, mientras que la encadenación hacia atrás intenta hallar datos para demostrar o refutar una hipótesis ya formulada. La encadenación hacia adelante pura conduce a un cuestionario descentrado del sistema del tipo "¿Y si...?", mientras que la encadenación hacia atrás pura suele ser inexorable con su interrogatorio dirigido a un objetivo.

La mayoría de los sistemas de éxito utiliza una mezcla de ambas estrategias. Si un procedimiento deductivo trabaja básicamente hacia atrás o hacia adelante, habrá de tratar con datos inciertos. Los especialistas en ordenadores han intentado simplificar y comprimir el mundo en que vivimos con el fin de delimitarlo y hacerlo compatible con los confines rígidos del ordenador, y este cometido nunca ha sido fácil. Ahora la investigación en sistemas expertos nos ha proporcionado un medio para tratar con lo incierto; en otras palabras, con el mundo real en vez de con alguna abstracción idealizada que nuestro sistema de datos nos obligue a utilizar.

En realidad tenemos demasiadas formas de tratar con lo incierto. Están la lógica bayesiana, la lógica polivalente y los factores de certidumbre, por nombrar sólo tres: estas lógicas reemplazan la certeza de "IF (SI) X ES VERDADERA, THEN (ENTONCES) Y ES VERDADERA" por la prudente deducción estadística de "IF (SI) X ES VERDADERA EL 65 % DE LAS VECES, THEN (ENTONCES) LAS PROBABILIDADES DE Y OSCILAN ENTRE EL 50 Y EL 70 %. Se han probado toda clase de esquemas, y lo curioso es que todos parecen funcionar. Una explicación para este hecho podría basarse en que la organización del conocimiento importa más que los valores a él asignados. La mayoría de las bases de conocimiento incorporan la redundancia para que el sistema experto llegue a conclusiones correctas por varios caminos.

Los paquetes de software diseñados para facilitar el acceso del usuario al diseño de sistemas basados en el conocimiento están apareciendo ya en el mercado. En Gran Bretaña se puede adquirir el HULK (*Helps Uncover Latent Knowledge*: ayuda a descubrir el conocimiento latente), que comercializa Brainstorm Computer Solutions. Sólo funciona en el BBC Modelo B y en los ordenadores Torch; no obstante, es muy probable que aparezca una versión para el QL, de Sinclair.

El HULK permite que el usuario elabore y verifique un conjunto de reglas de decisión, que después se pueden usar para predicción o clasificación.

Por ejemplo, supongamos que un agricultor ha realizado en un ordenador personal mediciones detalladas acerca de la altura, el color de las hojas, etc., de varios centenares de naranjos, registrando, además, aquellos árboles que sufrieron alguna afección antes de la época de la recolección y aquellos que se conservaron sanos. El sistema se podría utilizar para desarrollar reglas que relacionaran las características del árbol con su estado de salud. Luego esas reglas podrían identificar a los naranjos en peligro y, por consiguiente, mejorar las expectativas para la recolección del año siguiente. Este agricultor quizá no sabría nunca cuáles eran esas reglas, pero el sistema las aplicaría a los datos relativos a los árboles tal como se le proporcionaron.

Por otra parte, usted podría retener información acerca de los resultados de los partidos de fútbol de la temporada y desarrollar un procedimiento para catalogar los diferentes encuentros como probables triunfos, empates o derrotas, sobre la base de diversos indicadores conocidos antes del saque inicial. Los datos necesarios para tomar decisiones, como el rendimiento reciente del equipo local, los resultados previos de este encuentro, las posiciones de los equipos en la tabla de clasificación, todo está fácilmente disponible. Pero si no se contara con la ayuda de un SE se ocuparía mucho tiempo en organizar esta información y sería difícil correlacionarla. Existen muchas aplicaciones para el HULK: todo cuanto necesita es un conjunto de datos.

El paquete HULK consta de dos programas principales: el LOOK (*Logical Organiser of Knowledge*: organizador lógico de conocimiento) y el LEAP (*Likelihood Estimator and Predictor*: estimador y pronosticador de probabilidades). Éstos permiten que el usuario desarrolle un conjunto de reglas a partir de un archivo de datos de observaciones retenido en disco (p. ej., los resultados de los encuentros anteriores y otros factores de decisión) y luego aplicar ese conjunto de reglas a otro archivo de datos de observaciones incompletas (p. ej. los factores de decisión para los partidos del próximo domingo) con el objeto de hacer pronósticos respecto a los mismos. El conjunto de reglas es la base de conocimientos: expresa el conocimiento del sistema acerca de los datos. En el HULK, consiste en reglas de decisión en cuanto a lo probable, que se pueden usar para clasificación o predicción.

La base de conocimientos crece a través de la interacción del usuario y el ordenador: las reglas las propone el usuario y las verifica el ordenador; el usuario descarta aquellas que no mejoran el rendimiento global del sistema.

Para utilizar el LOOK uno necesita un conjunto de datos o, mejor aún, dos (un gran conjunto de datos se puede dividir en dos partes con este fin). Uno se denomina *conjunto de entrenamiento* para probar las reglas, y el otro recibe el nombre de *conjunto de prueba* para confirmar su utilidad sobre datos no vistos.

Una vez los datos están en el archivo, se emplea el LOOK para probar sus ocurrencias mediante la proposición de nuevas reglas, de a una cada vez. Hace pasar cada regla por el conjunto de entrenamiento y le dice en cuánto mejora (si es que lo mejora) el marcador de predicciones de las reglas ya presentes. Luego aconseja si es conveniente conservar o descartar la nueva regla. La decisión final queda en manos del usuario.

Después de crear las reglas mediante el LOOK, se puede utilizar el LEAP para aplicarlas a muestras desconocidas. Las reglas se combinan para dar una estimación de una única probabilidad para cada muestra del conjunto de datos de prueba. La salida del LEAP incluye una lista de muestras ordenadas de acuerdo a su probable resultado. Lo que pueda ser este resultado (MARCADOR EMPATE, o ALTO RENDIMIENTO DE ESTE NARANJO) depende de la naturaleza de los datos de muestra y de lo que el usuario intentaba pronosticar mediante los mismos. El HULK proporcionará la configuración del módulo de adquisición y la base de conocimientos, dejándole al usuario el control del motor de deducción.

Ampliación flexible

El sistema de ampliación del BBC Micro ha mejorado aún más gracias a que la Acorn ha diseñado un nuevo DOS y un controlador para unidades de disco

En la actualidad, virtualmente todas las unidades de disco compatibles con el BBC Micro son minifloppies estándar de 5 ¼ pulgadas, unidades de capacidad simple o doble, pero están apareciendo unidades de 8 pulgadas. Lamentablemente, el DFS no es una pieza estándar en los ordenadores BBC y se debe comprar por separado a Acorn. Se vende en forma de un chip de ROM que encaja en un conector del tablero de circuito impreso del ordenador. Otros fabricantes ofrecen chips DFS alternativos por un precio algo inferior al de Acorn.

Debido a que el DFS está contenido en ROM, la mayoría de las órdenes de disco no requieren memoria interna para llevar a cabo las instrucciones. El DFS se amplía mediante órdenes y rutinas proporcionadas como "utilidades" en un disco que viene con la unidad.



Exclusiva y automática
La Hobbit constituye un sistema exclusivo de cinta flexible diseñado para el BBC Micro. Al estar completamente controlada por software, todas las funciones de avance, rebobinado, reproducción y grabación se realizan de forma automática

Las unidades de disco se acoplan a través de un cable plano a un conector de 34 vías (que lleva la indicación *disk drive*) en la cara inferior del ordenador. Este cable transporta todos los datos del disco a y desde las unidades, tanto dobles como simples. El control de las unidades de disco se ejerce a través de un chip controlador de disco 8271 que convierte los datos de ocho bits en paralelo del ordenador a la forma en serie para salir a la unidad de disco seleccionada, y viceversa. Se admiten cuatro estándares de formato: 40 pistas u 80 pistas en una sola cara y 40 u 80 pistas en doble cara, siempre a densidad simple. Cada pista se divide en 10 sectores, que contiene cada uno 256 bytes, dando una capacidad total de unos 100 Kbytes por cara en un formato de 40 pistas y 200 Kbytes por cara en el de 80 pistas.

Las unidades de disco BBC
Para sacar el máximo partido de un BBC Micro es esencial una unidad de disco. Las dos aquí ilustradas son de las más populares para el Modelo B: la Acorn 100 y la Torch Z80. Para poderlas utilizar, primero se deben conectar en interface con el ordenador a través de unidades ROM de sistema operativo en disco, que se instalan en el interior de la máquina

Ian McKinnell

Las unidades se identifican mediante un número; una única unidad es 0 y una segunda unidad es 1. En el caso de las unidades que emplean las dos caras de un disco, ambas caras se tratan como unidades separadas, numeradas 0 y 2. Una segunda unidad numerará sus dos caras 1 y 3.

El catálogo del disco (o cara) ocupa dos Kbytes y está contenido en los dos primeros sectores de la primera pista. Dicho catálogo retiene información acerca del nombre y los identificadores del disco y se puede dividir en un máximo de 27 directorios seleccionables independientes con las listas de los archivos correspondientes, de manera que estos últimos se pueden almacenar por categoría. No hay previsto ningún mapa de disponibilidad de bloques (BAM); en cambio, hay una orden *COMPACT que localiza cualquier hueco que hubiera dejado algún archivo borrado y reacomoda el almacenamiento de los archivos por orden, dejando todo espacio libre después del final del último archivo.

Los archivos de programas y de datos se pueden guardar en disco del mismo modo que en cinta. De hecho, con el DFS todas las órdenes para manipulación de cassettes se dirigen automáticamente al disco a partir del encendido. Para utilizar una cassette, con el DFS acoplado, dé entrada a la orden RETURN — * TAPE. Para reasignar el disco dé entrada a: RETURN — * DISK. Además de las órdenes para manipulación de archivos estándar, el DFS facilita órdenes y rutinas para manipulación de datos, incluido un sistema archivador de acceso aleatorio, órdenes HELP y mensajes de error.

Al igual que el excelente BASIC BBC, el DFS es un poderoso dispositivo para manipular datos. Incluye muchas rutinas que por lo general se consideran fuera del alcance de los sistemas operativos en disco para ordenadores personales y, además de ello, son fáciles de utilizar. La estructura del sistema fomenta la administración eficaz del disco y la transferencia de datos es rápida. Las cifras varían de un fabricante a otro, aunque, por término medio, la carga de un archivo de programas de 20 Kbytes tarda alrededor de cinco segundos. El inconveniente del sistema, aparte de su costo relativamente elevado, es que el DFS sólo puede almacenar 31 nombres de archivo para cada cara de disco. Habida cuenta de la posible capacidad de 200 Kbytes por cara y la flexibilidad que caracteriza al sistema de directorio, esto representa un auténtico inconveniente.



Órdenes del disco BBC

Siempre que sea necesario enunciar el archivo con el cual esté relacionada una orden, la especificación completa es:

COMMAND:DV.DR.NOMARCHI

donde COMMAND es la orden para manipulación de archivos requerida; DV es el número de unidad de disco (0-3); DR es el identificador de directorio (A-X o $); y NOMARCHI consta de un máximo de siete caracteres que identifican el archivo (no se deben incluir '#', '*', '.' ni ':'). De no especificarse el número de unidad ni el identificador de directorio, la unidad por omisión es 0 y el directorio por omisión es $. Los identificadores por omisión se pueden cambiar utilizando las órdenes *DRIVE, *DIR & *LIB. El DFS permite asimismo el empleo de estos caracteres "wildcard": '#' y '*'. En algunas órdenes éstos se pueden utilizar para especificar todas las unidades, todos los directorios o todos los nombres de archivo que empiecen con la misma letra. En las explicaciones que proporcionamos, la anterior especificación para archivos se abrevia como <FSP>. Además de las órdenes para cassette estándar disponibles para el disco desde el momento del encendido, y de las órdenes de acceso directo, el DFS facilita estas órdenes para manipulación de discos:

***FORM40, *FORM80 y *VERIFY**
Estas órdenes están almacenadas como utilidades en un disco que se proporciona con la(s) unidad(es) de disco; dan formato a un disco de 40 u 80 pistas e informan acerca del éxito de la operación.

ACCESS
*ACCESS <FSP> L protege al archivo de un borrado o sobreescritura. *ACCESS <FSP> anula esta protección.

***BACKUP, *DESTROY y *ENABLE**
*BACKUP FuenteDV Destino DV copia el contenido completo del disco de la unidad fuente en el disco de la unidad de destino, sobreescribiendo, por consiguiente, el disco de destino. *DESTROY <FSP> elimina un archivo. De haber *wildcards* (tarjetas ajenas) incluidas en <FSP>, mediante una sola orden se podrían borrar varios archivos. Dado que BACKUP y DESTROY son tan drásticas en cuanto a sus efectos, se debe dictar una orden *ENABLE para que el DFS las obedezca.

***BUILD <FSP>**
Crea un archivo ASCII con la especificación de archivo a partir de todas las subsiguientes entradas desde el teclado hasta acabar con la tecla ESCAPE.

***CAT DV**
Visualiza el catálogo de la unidad de disco especificada.

***COMPACT DV**
Desplaza los espacios libres de un disco de la unidad especificada hasta el final del último archivo, en un bloque continuo.

***COPY**
Copia el o los archivos especificados (utilizando un nombre de archivo wildcard) de un disco a otro.

***DELETE <FSP>**
Borra el archivo individual especificado del catálogo de un disco. El archivo puede sobreescribirse después.

***DIR DR**
Establece el directorio por omisión corriente en el directorio especificado. Todos los archivos subsiguientes almacenados mediante *SAVE o SAVE se le asignarán al directorio establecido.

***DRIVE DV**
Establece la unidad por omisión corriente.

***DUMP <FSP>**
Visualiza un listado en hexadecimal del archivo especificado.

***EXEC <FSP>**
Lee todos los datos de un archivo como si a éste se le hubiera dado entrada desde el teclado. Es útil para ejecutar una secuencia de órdenes que se emplee con frecuencia. Los archivos que se pueden leer mediante *EXEC se crean mediante *BUILD.

***HELP**
Referida a la operación de la unidad de disco, *HELP DFS visualiza una lista parcial de las órdenes DFS estándar y de su construcción, y *HELP UTILS visualiza una lista de las demás órdenes DFS estándar.

***INFO <FSP>**
Visualiza información extra relativa al o a los archivos especificados (utilizando *wildcards*) no visualizados mediante *CAT, tales como: posición de memoria, dirección de ejecución, longitud en bytes y localización de sector.

***LIB:DV.DR**
Establece el directorio especificado como la "biblioteca" (en inglés, *library*). Permite la utilización de una forma breve de orden (*NOMARCHI) que busca el directorio corriente de la biblioteca para el programa en lenguaje máquina mencionado, lo carga en la memoria y lo ejecuta inmediatamente como si se hubiera empleado la orden *RUN completa.

***LIST <FSP>**
Visualiza el archivo ASCII especificado, incluyendo los números de línea.

***LOAD <FSP>**
Lee el archivo especificado en la memoria, en las posiciones de las que se tomara originalmente.

***OPT 1**
Instrumenta un sistema de mensajes en el que se visualiza cada vez que se accede a un archivo la información dada por *INFO. Esta facilidad se logra mediante *OPT 1 1. Para inhabilitar esta configuración, utilice *OPT 1 0.

***OPT 4**
Modifica la opción de comienzo automático al encenderse o con (SHIFT) BREAK para la unidad de disco corriente seleccionada, donde: *OPT 4 0 desactiva el arranque automático; *OPT 4 1 carga (LOAD) el archivo !BOOT; *OPT 4 2 ejecuta (RUN) !BOOT; y *OPT 4 4 ejecuta (EXEC) !BOOT.

***RENAME <FSP viejo> <FSP nuevo>**
Esta orden modifica el nombre de un archivo y lo desplaza a un directorio diferente. No desplaza archivos de una unidad a otra.

***RUN <FSP>**
Lee un archivo en código de lenguaje máquina en la memoria y lo ejecuta inmediatamente. Se utiliza con archivos que no estén contenidos en la biblioteca corriente.

***SAVE**
Copia un bloque especificado de la memoria del ordenador y lo escribe en disco en la unidad y el directorio corrientes. Se construye así:

*SAVE "NOMBRE" CCCC FFFF EEEE RRRR

o bien

*SAVE "NOMBRE" CCCC + LLLL EEEE RRRR

donde CCCC es la dirección de comienzo del bloque de memoria; FFFF es la dirección de final del bloque de memoria; EEEE es la dirección de ejecución del programa almacenado; RRRR es la dirección a la cual se le leerá el programa; y LLLL es la longitud del archivo expresada en bytes (opción para FFFF). Todos los números se dan en hexadecimal. RRRR y EEEE se pueden omitir, y las direcciones de recarga y ejecución pasan, por omisión, a CCCC.

***SPOOL <FSP>**
Abre el archivo especificado para recibir toda la información visualizada como un archivo de textos. Permite almacenar un programa en BASIC como un archivo ASCII, en lugar de distintivarlo.

***TITLE "NOMBRE DISCO"**
Cambia el nombre del disco en la unidad corriente por el nombre especificado.

***TYPE <FSP>**
Visualiza un archivo ASCII excluyendo los números de línea.

***WIPE <FSP>**
Es idéntica a *DESTROY, con la excepción de que *ENABLE no es necesaria.

La gran prueba

La etapa final de toda operación de ensamblaje es ponerlo a prueba, pero esto no incluye conectarlo, pues podría destruirse uno de los dispositivos más delicados

El primer paso para probar cualquier ensamblaje de componentes en un circuito en funcionamiento se refiere a la eficacia de las junturas soldadas. Una juntura efectuada a una temperatura demasiado baja puede parecer aceptable exteriormente, pero por dentro quizá no establezca conexión alguna. Un suave tirón permitirá comprobar adecuadamente este extremo: en el caso de que la juntura esté realizada de forma correcta no se le separará entre las manos; por otra parte, en el supuesto de que se separe, es mejor que la juntura fracase en esta etapa que más adelante.

Ahora estamos preparados para aplicar algún *tester*. Uno de los más sencillos se puede comprar en las ferreterías o tiendas de recambios para automóviles. Se emplean para determinar cuándo los puntos de contacto del distribuidor están abiertos y cerrados.

Sin embargo, una alternativa mejor es la que representa un pequeño multímetro. En su papel de ohmiómetro puede probar no sólo la continuidad sino también la resistencia. La unidad de resistencia se denomina *ohmio* en honor del físico alemán del siglo XIX Georg Ohm (1789-1854), que fue quien descubrió el fenómeno. La resistencia es una función de la superficie de la sección transversal de un cable que transporta una corriente, pero también se puede crear artificialmente mediante la utilización de unos componentes llamados *resistencias*. Para nuestra tarea, una juntura efectuada correctamente ofrecerá una resistencia despreciablemente pequeña al paso de la pequeña corriente empleada por el medidor o *tester* de continuidad, y, como resultado de ello, la luz se encenderá y brillará con intensidad, o bien, la aguja del dial se ladeará por completo.

Toda reacción inferior indica que la juntura es pobre y se debe realizar otra vez.

Además de medir la conductividad, el multímetro posee otras dos funciones: la medición de la corriente en amperios y del potencial eléctrico en voltios. Estas dos unidades guardan una estrecha relación entre sí: en efecto, la diferencia de potencial existente entre dos puntos de un circuito que transporta un amperio de corriente y gasta un vatio de energía es un voltio.

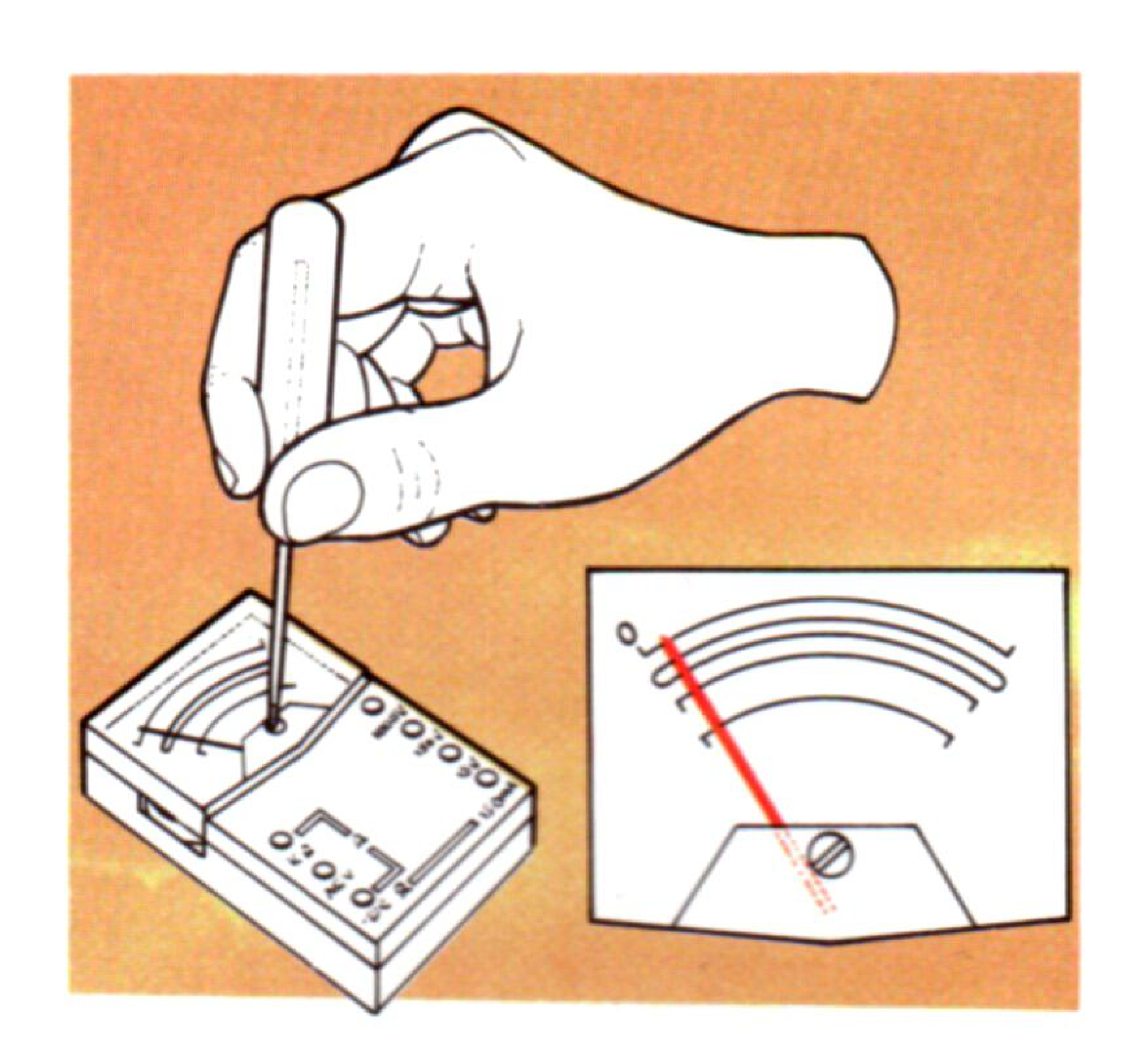
Multímetro

SECTOR REFLEJADO
ESCALA DE RESISTENCIA
TORNILLO DE PUESTA A CERO
TERMINALES COMUNES Y DE CONTINUIDAD
TERMINALES POSITIVOS
CABLES
KΩ
ACV
-DC Ω
Ω
1000V
150V
15V
150mA

Típicamente, los multímetros prueban la continuidad y la resistencia, medida en ohmios; la magnitud del flujo de corriente, medida en amperios (amps) y la cantidad de potencial (en ocasiones llamada *tensión*), medida en voltios. El precio de los multímetros varía. No obstante, sólo hay dos métodos de representar sus resultados: analógico o digital. En general, los instrumentos de bobina móvil que visualizan sus datos mediante el movimiento (deflección) de una aguja a través de una escala de forma análoga al aumento o la disminución del valor que se está midiendo, son considerablemente más baratos que las versiones digitales

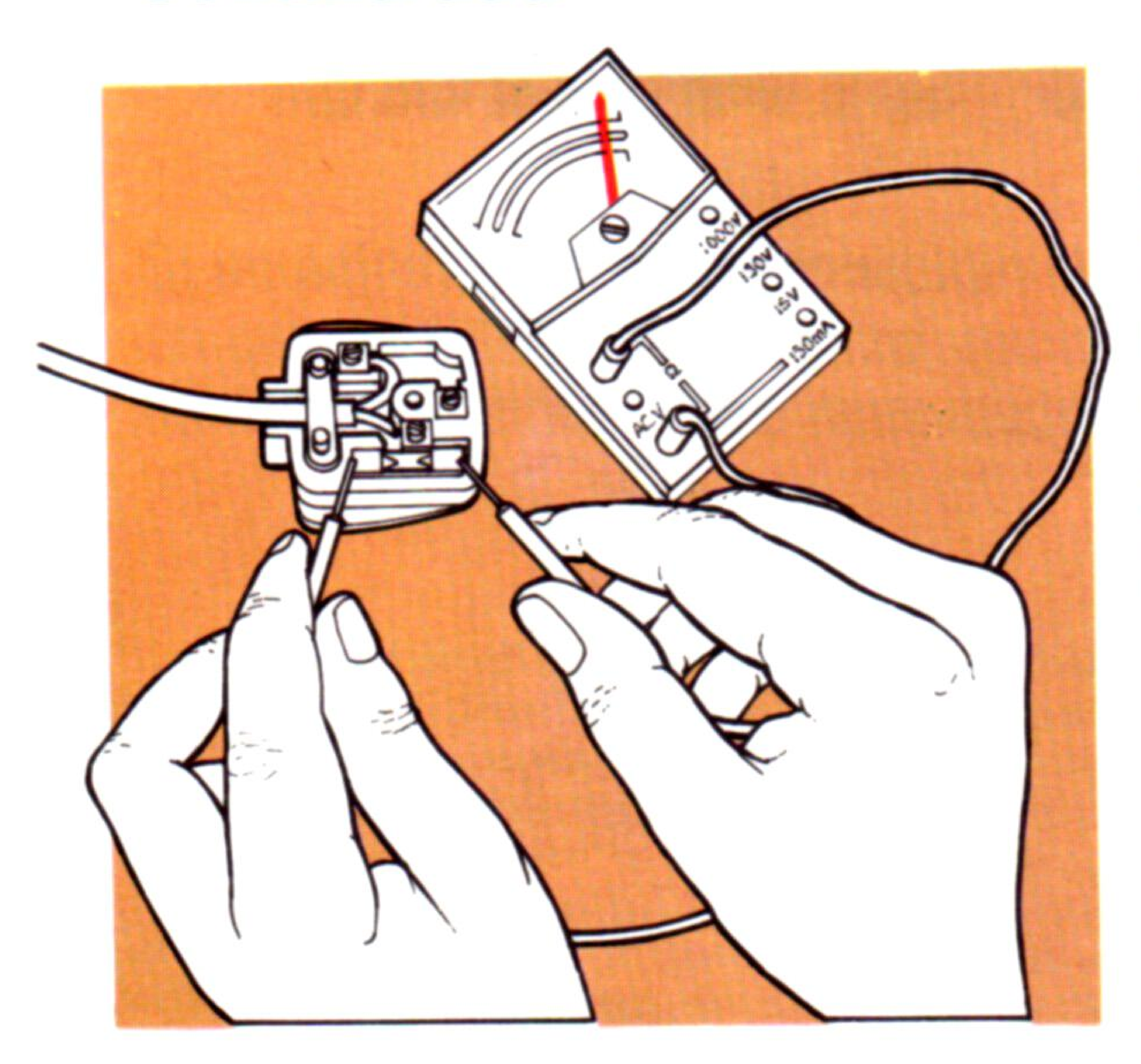
Tester de circuitos

EXTREMO
PILAS DE 1,5 VOLTIOS
BOMBILLA INDICADORA
PINZAS COCODRILO

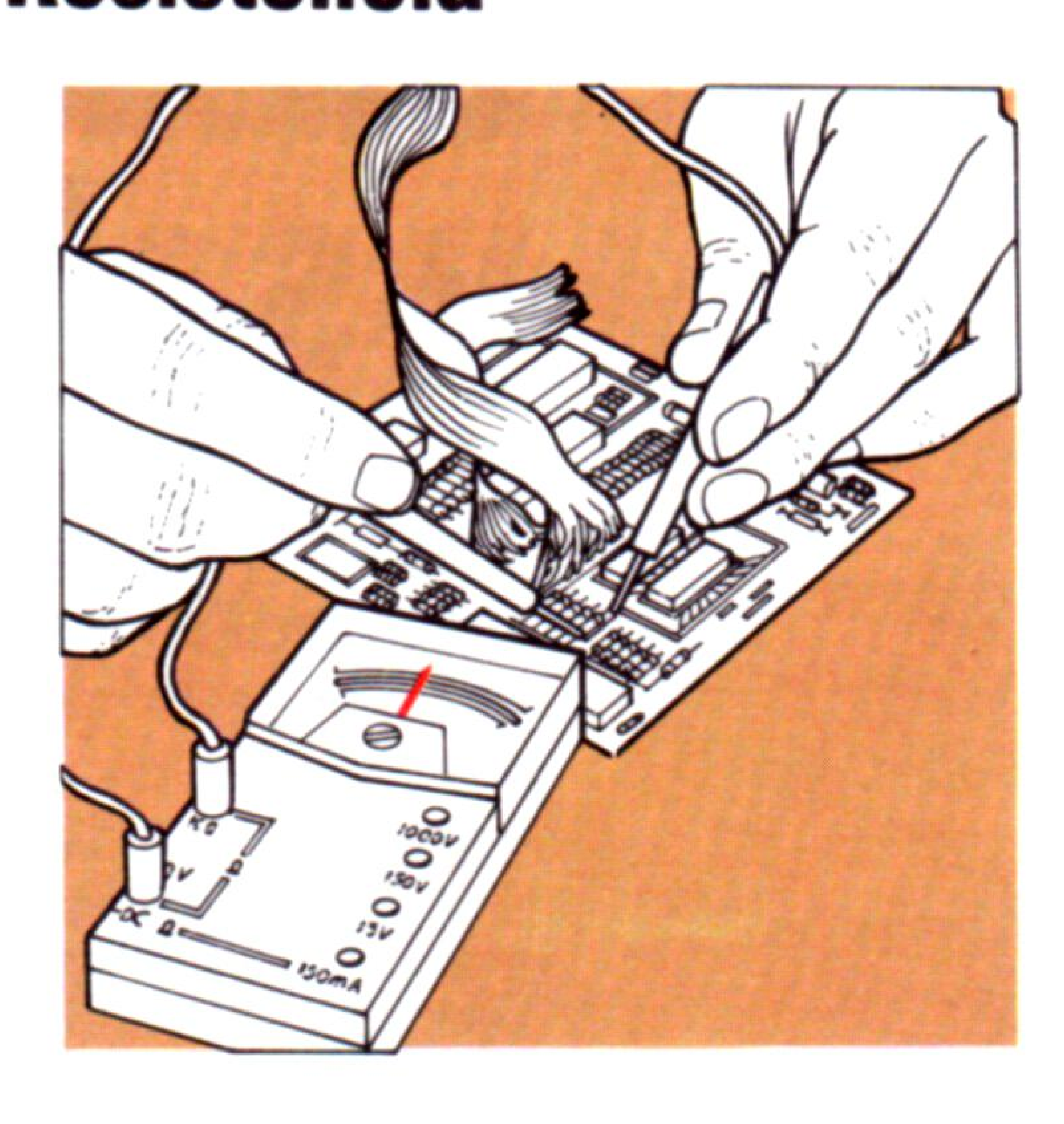


El medidor puesto a cero
Los medidores analógicos, que requieren que la aguja indicadora se mueva físicamente a través de un dial, deben disponer de algún medio de ajuste. Normalmente asume la forma de un tornillo, montado en el punto de apoyo de la aguja. Un sector reflejado por detrás de la aguja le permite al observador estar seguro de una lectura exacta. Además, el circuito de medición de resistencia también debe ser ajustable, para registrar otras variaciones e imprecisiones

Continuidad

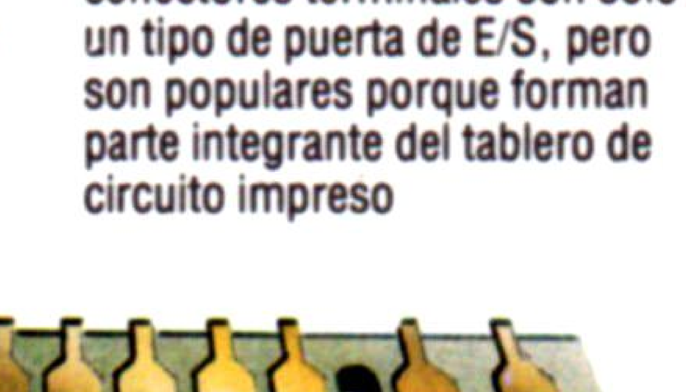


Resistencia

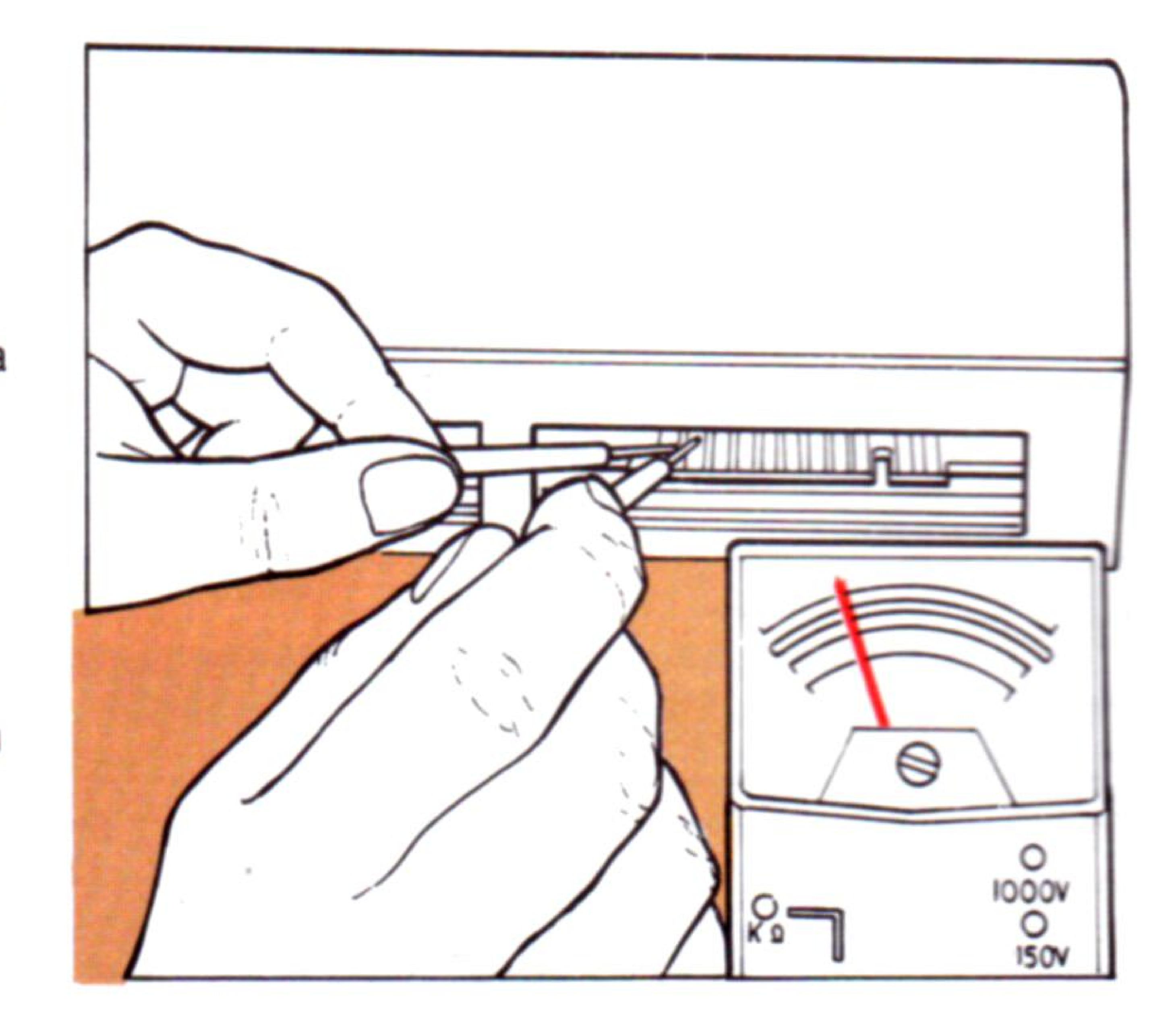


Continuidad y resistencia
Uno de los ejemplos más comunes de fallo de continuidad inducido y deliberado es el del fusible existente en todos nuestros enchufes de 13 amperios con tres patillas. El fusible está diseñado para quemarse y romper el circuito si se llegara a sobrecargar y, en ausencia de un *tester* de continuidad, el remedio normal cuando un aparato no funciona consiste en sustituir el fusible viejo por uno nuevo. Pero, ¿y si no era el fusible? El *tester* de continuidad nos lo dirá enseguida. Además, podemos utilizar un multímetro para medir el valor de una resistencia

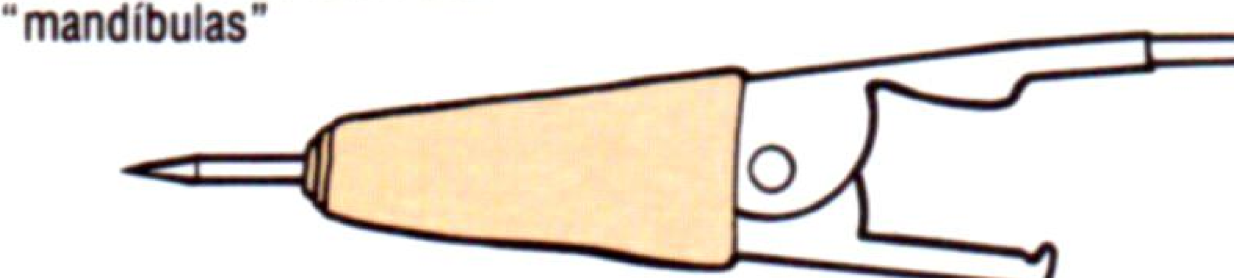
Flanco de subida
Los conectores terminales como el que vemos en la ilustración son el medio físico mediante el cual el ordenador se conecta con sus periféricos. Algunas máquinas, como el Spectrum y el ZX81, poseen una sola de estas puertas. Otras, como el BBC Micro, poseen varias. Los conectores terminales son sólo un tipo de puerta de E/S, pero son populares porque forman parte integrante del tablero de circuito impreso

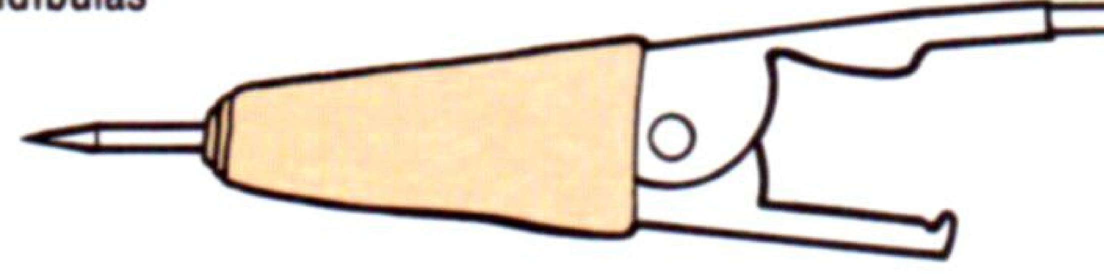


Voltaje
Un buen ejemplo del empleo del multímetro como *tester* de voltaje consiste en localizar la fuente de alimentación eléctrica del conector marginal de su ordenador. Consulte su manual y ponga la patilla de 0 voltios y la de +5 o +9 voltios. Ponga en contacto el cable negativo del medidor con 0 voltios y el positivo con +5 o +9 voltios. La aguja le señalará con exactitud cómo están la regulación de voltaje y el sistema de circuitos de control del interior de su ordenador y su fuente de alimentación eléctrica



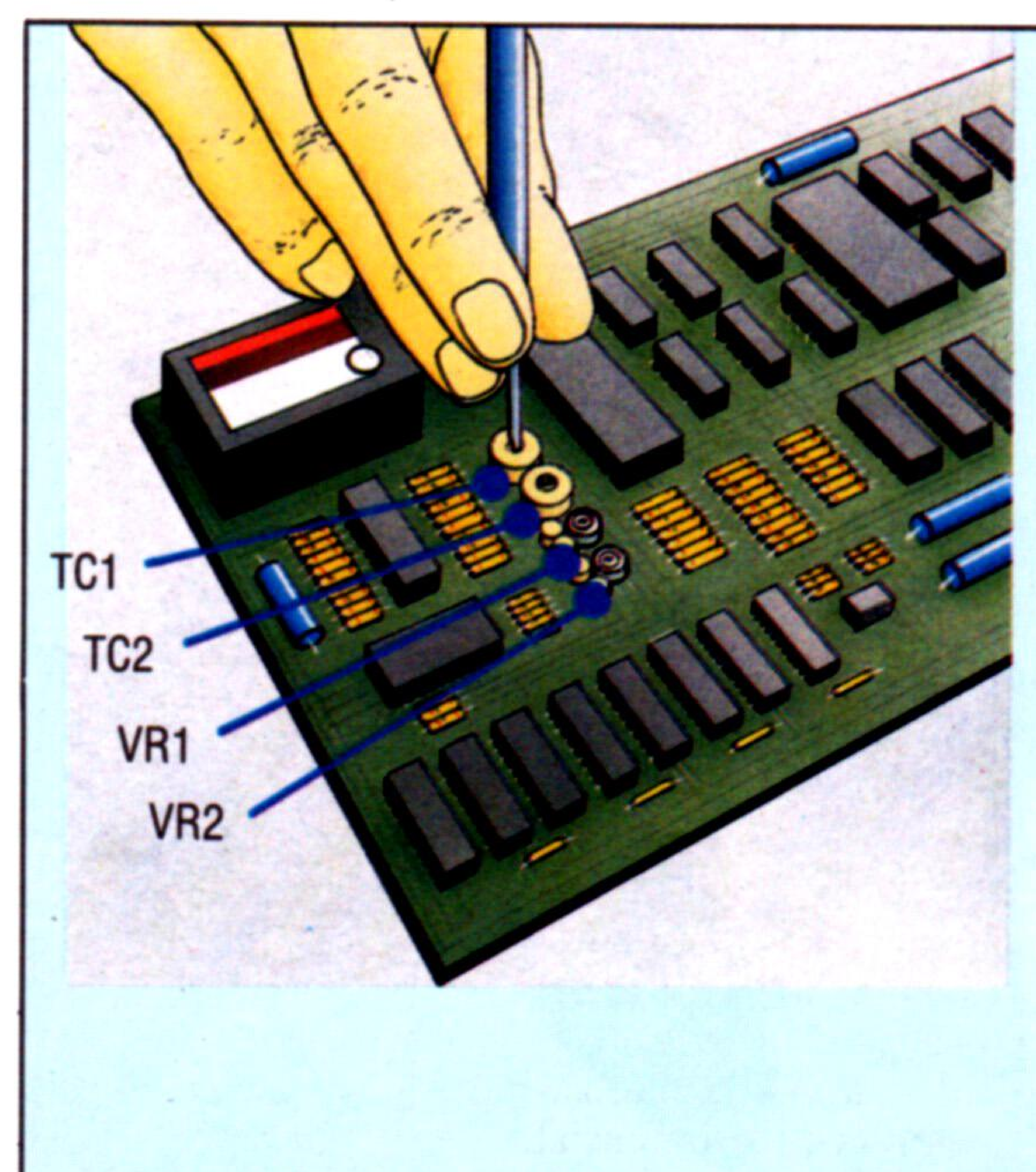
Pinzas cocodrilo
Las pinzas cocodrilo, bastante toscas, se pueden convertir en un cable más refinado pegando con cinta aislante un trozo de hierro de soldar entre sus "mandíbulas"





ATENCIÓN

la garantía de su ordenador personal (en caso de que aún estuviera en vigor) podría anularse si alguien que no fuera el fabricante o su agente autorizado abriera la carcasa



Sintonice su Spectrum

Como observábamos en la p. 530, los usuarios de los primeros Spectrum pueden mejorar la pobre calidad de imagen en el televisor. Las dos resistencias variables que vemos (VR1 y VR2) controlan respectivamente el equilibrio rojo-verde y azul-amarillo. TC1 (un condensador de ajuste) y su compañero TC2 controlan la claridad de los caracteres y la intensidad del color.
La carcasa del Spectrum está sujeta con cinco tornillos estrella visibles situados en la cara inferior (tome nota del aviso sobre la caducidad de la garantía); para dejar al descubierto el tablero basta con quitar estos tornillos.
Los usuarios de la versión 3 del Spectrum no pueden realizar este ajuste ni ningún otro. Un Spectrum versión 3 se puede identificar por el disipador, placa de aluminio visible a la izquierda del conector terminal

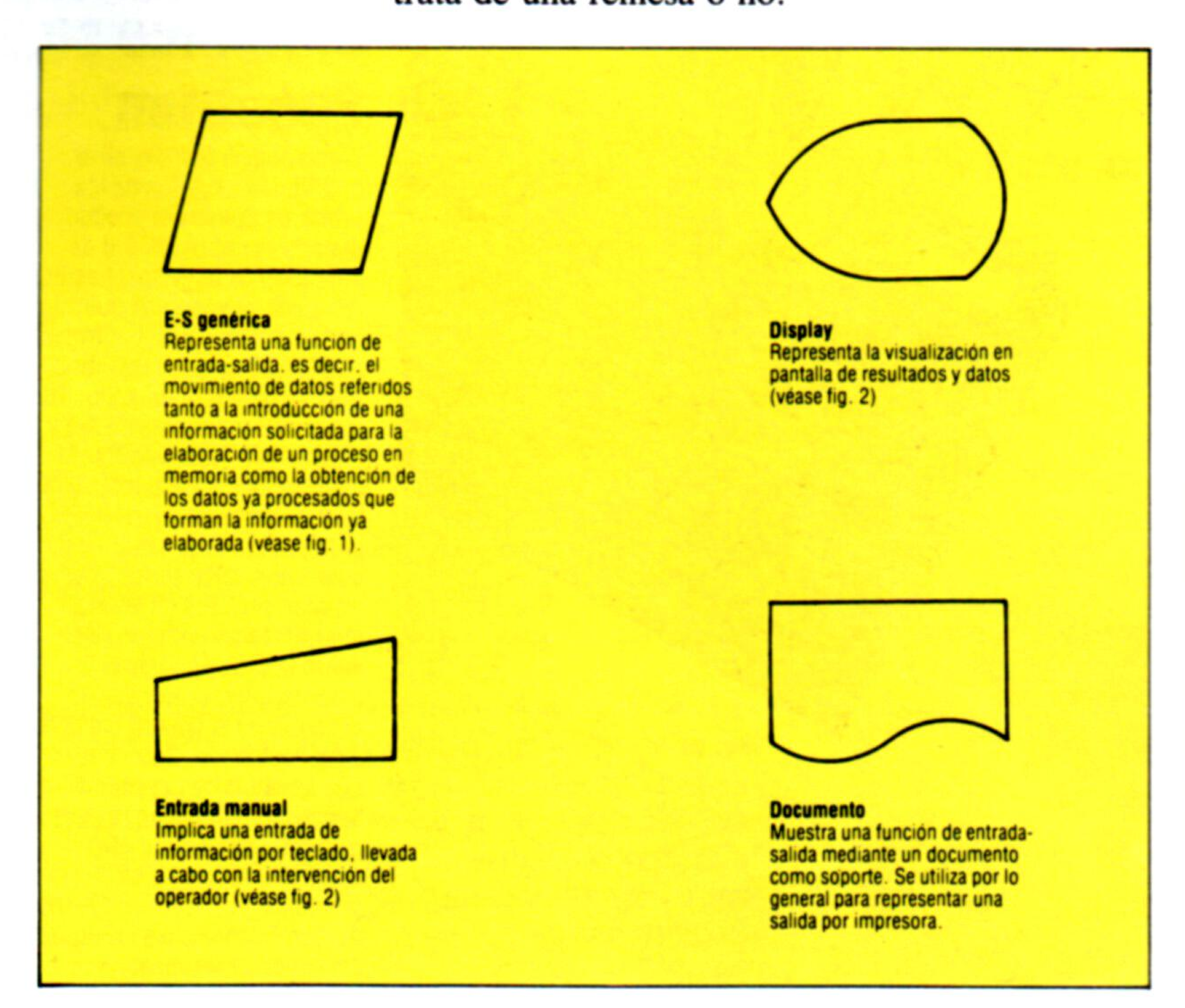
Símbolos de entrada y salida

Entenderemos su uso mediante un sencillo ejemplo contable

Supongamos el siguiente ejemplo: "Los clientes de una firma ven modificadas sus fichas (donde se guardan datos referidos tanto a razones sociales como a movimientos de géneros, pagos, etc.), conforme a los respectivos comprobantes, que obran en poder de las empresas y son diariamente timbrados. En ellos aparece la cantidad de género en movimiento que, si se refiere a una remesa, se deberá sumar a la existencia de la ficha correspondiente, mientras que si se trata de una devolución, la existencia se restará. En cualquier caso se mostrará la ficha como final de proceso, debidamente actualizada".

Una vez analizado el problema, comprobamos que un tipo de operación aparece repetido, ya que podemos distinguir dos entradas diferentes: la de la ficha correspondiente al cliente y la del comprobante. Dado que no se informa expresamente de la naturaleza de dichas entradas, éstas se considerarán genéricas.

Así, cuando leemos *tomar comprobante*, se entenderá por ello dar la cifra que en él aparezca, o sea, la cantidad correspondiente a la existencia en el caso de la ficha del cliente. Por otra parte, en la fase de *salida*, se pide que aparezca la nueva configuración de la ficha debidamente actualizada tras el proceso que tiene lugar una vez se ha decidido si se trata de una remesa o no.



E-S genérica
Representa una función de entrada-salida, es decir, el movimiento de datos referidos tanto a la introducción de una información solicitada para la elaboración de un proceso en memoria como la obtención de los datos ya procesados que forman la información ya elaborada (véase fig. 1).

Display
Representa la visualización en pantalla de resultados y datos (véase fig. 2)

Entrada manual
Implica una entrada de información por teclado, llevada a cabo con la intervención del operador (véase fig. 2)

Documento
Muestra una función de entrada-salida mediante un documento como soporte. Se utiliza por lo general para representar una salida por impresora.

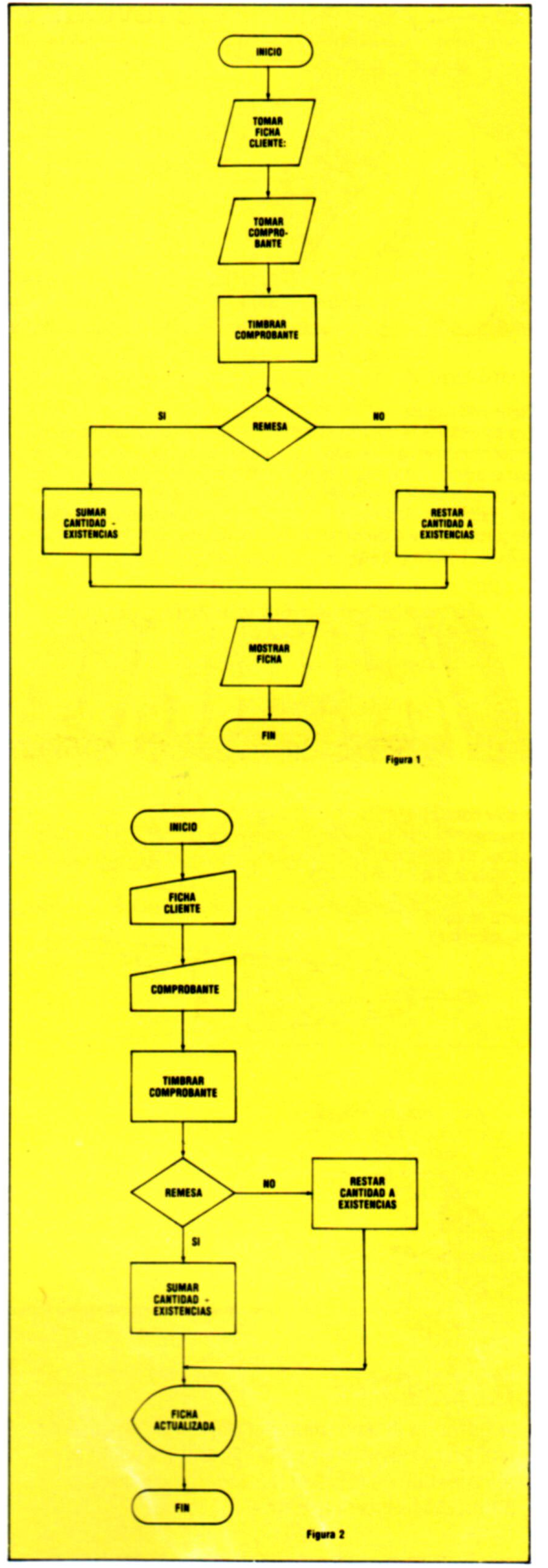


Figura 1

Figura 2



Informática de calidad

Si se afirma que la revolución del microordenador empezó el día en que la IBM entró en liza, recordemos que su entrada, en 1981, es muy reciente. Pero revolucionó el mercado

La reputación de IBM en los mercados del miniordenador y del ordenador de unidad principal no es, por cierto, la de una empresa innovadora en cuanto a hardware. Su software y su documentación, aunque con frecuencia profusamente detallados y exactos, se podrían mejorar en términos de presentación y sencillez de uso. Sin embargo, la empresa es celebrada por la solidez de sus equipos y el IBM PC es, ciertamente, robusto. Al igual que casi todo el hardware IBM, cuesta también considerablemente más que sus competidores.

Lo cual no parece ser un factor negativo a la hora de las ventas, pues, a pesar de unos precios que hacen parecer baratos a muchos ordenadores, la máquina se convirtió enseguida en una de las más populares. Se le ha hecho el definitivo cumplido de ser copiada e imitada al menos tanto como el Apple y, por cierto, más rápidamente.

IBM explica que el elevado precio del PC es un reflejo del nivel de asistencia que ofrece la empresa. Dicha asistencia existe en realidad, si usted está dispuesto a pagar el 11,2 % del costo del producto por año para mantener un contrato de servicio. El costo de la mayoría de los contratos de servicio independientes es superior al menos en un 2 % y pocas empresas pueden ofrecer unidades de recambio en cuestión de momentos, de modo que tal vez exista algún mérito adicional en adquirir máquinas construidas por una empresa tan grande como IBM.

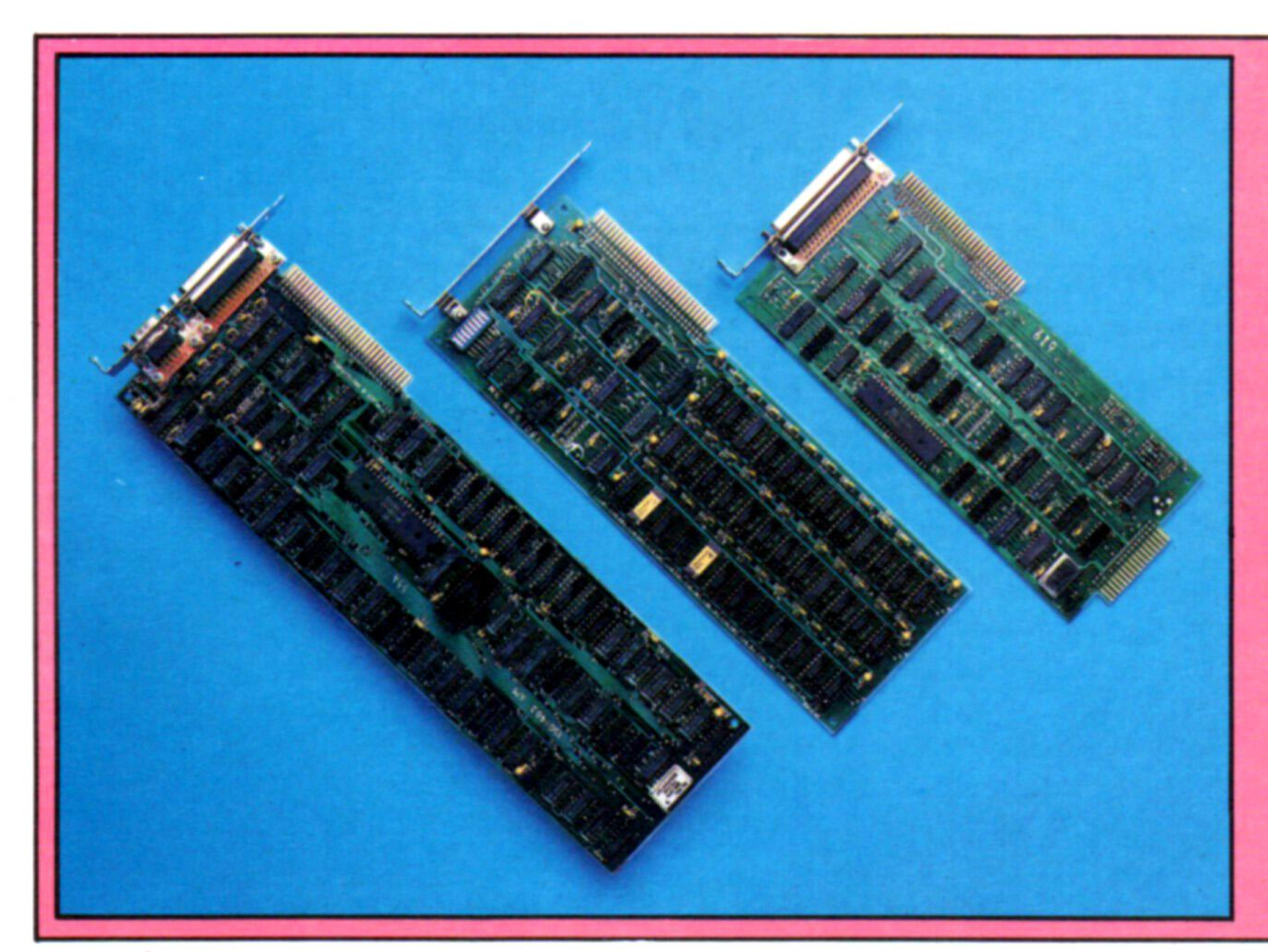
Las especificaciones del IBM PC no son descollantes. Posee un procesador 8088, descrito como una CPU de 16 bits, pero tiene las líneas de dirección y de datos multiplexadas para economizar patillas del chip y esto significa que no es rápida. De hecho, su rendimiento suele ser sólo alrededor del 25 % más rápido que una mediana máquina de ocho bits.

Tal como se suministra, el modelo básico necesita una ampliación para aprovechar al máximo su potencial, porque no posee mucha memoria (no la suficiente para ejecutar programas complicados) y casi no posee facilidades de entrada y salida. En



Chris Stevens

Diseño ergonómico
El diseño físico y el trazado del IBM PC reflejan la dilatada experiencia de la empresa en el campo de ordenadores y productos para oficina: son discretos y ergonómicamente sólidos. Las tres unidades principales (teclado, procesador y monitor) están separadas para facilitar su acomodamiento. La máquina viene con una unidad monocromática como estándar, pero existe a la venta un monitor en color



Tableros de ampliación
En su forma más elemental, el IBM PC apenas si constituye un desafío para aquellas máquinas que, como el Apple II, llevan mucho más tiempo establecidas en el mercado; pero cuando se amplía (de izq. a der.) ya sea con un tablero para gráficos en color, o con un tablero para ampliación de memoria o bien un sofisticado controlador de entrada-salida, la máquina comienza a parecerse mucho más a un serio ordenador personal para gestión empresarial

Unidades de disco
La máquina está equipada con una sola unidad de disco, de una sola cara y de densidad simple, pero ésta se puede mejorar progresivamente hasta una doble cara de densidad doble

consecuencia, la mayoría de los compradores se encuentran con que deben agregar al menos una ficha multifunción, que cuesta tanto como muchos ordenadores personales.

Los gráficos son impresionantes pero no se suministran con la máquina; el usuario necesita una ficha para gráficos. Ésta se vende en dos versiones (monocromática y en color) y se compone de un gran bloque de memoria (para retener la imagen de pantalla) y de componenentes electrónicos que producen la señal de video. Las fichas las controla la CPU principal, si bien está surgiendo un nuevo tipo de ficha de visualización que posee un procesador de visualización en video especializado. Éste libera al procesador principal de la tarea de actualizar las pantallas y, por consiguiente, acelera el proceso. Otra vez es de lamentar que sea tan caro.

El PC posee ranuras de ampliación, pero sólo hay cinco y, por tanto, se ha de pensar cuidadosamente en cómo se han de utilizar. El resultado es que virtualmente no existen a la venta fichas "baratas y geniales" que ofrezcan funciones limitadas a un precio reducido; casi todas ellas son grandes, complejas y capaces de realizar muchas tareas diferentes, a menudo de forma simultánea, y son caras. Dado que el IBM PC no sirve para mucho sin dichas mejoras, su costo se debería tener en cuenta a la hora de considerar la posibilidad de adquirirlo.

Existen, por supuesto, modelos alternativos del PC (como la versión XT de disco rígido) que proporcionan muchas de estas facilidades como estándar, pero los precios son mucho más elevados. En realidad se pueden comparar con el del Lisa de Apple, una máquina mucho más avanzada.

Como muestra de que ni siquiera IBM es inmune a las tendencias de la moda, el PC ha sido remodelado como un "portátil", pero con un peso mínimo de 14 kg este término resulta bastante forzado. La remodelación implicó reemplazar las dos unidades de disco de altura estándar por la unidad de doble cara, que deja entonces una de las aperturas disponible para un monitor de 9 pulgadas. Una versión del teclado más ligera y más pequeña se ajusta frontalmente a la máquina y se aloja en una nueva carcasa.

Tableros controladores de disco



Software IBM PC
Una de las principales razones para comprar un IBM PC (que es la misma por la cual tantos fabricantes de ordenadores de todo el mundo lo han copiado casi exactamente) es la selección de software comercial disponible. Éste funciona bajo el control del PC-DOS (sistema operativo en disco), desarrollado por Microsoft sobre la base del CP/M, aunque existe un número de alternativas (el CP/M-86 y UCSD p-system, p. ej.) y entre ellos estos sistemas operativos admiten una amplia variedad de lenguajes, como COBOL, FORTRAN, PASCAL y BASIC. La gama de software comercial es tan amplia como la de cualquiera de los sistemas de microordenadores existentes, incluyendo óptimos paquetes de procesadores de textos, hojas electrónicas, bases de datos y de gestión. Además, dado que en Estados Unidos el ordenador personal IBM PC se tiene en mucha más estima que en el resto del mundo, se dispone de un gran número de juegos producidos por empresas de software norteamericanas

Fuente de alimentación eléctrica





RAM
El IBM PC está equipado con 64 Kbytes de RAM, ampliable mediante otro medio megabyte a través de una ranura de ampliación

ROM

Ranuras de ampliación
Hay cinco ranuras de ampliación, pero tres de ellas están ocupadas por tableros esenciales, de modo que la capacidad de ampliación es limitada

Microprocesador
El IBM PC utiliza el microprocesador 8088 de Intel, que posee un direccionamiento de 16 bits pero una transferencia de datos de sólo 8 bits

IBM PC

DIMENSIONES

140 × 500 × 400 mm

CPU

Intel 8088

CAPACIDAD DE MEMORIA Y VELOCIDAD

64 K de RAM, ampliables a 576
40 K de ROM 4,7 MHz

CARACTERISTICAS DE LA PANTALLA

25 líneas de 80 caracteres

INTERFACES Y PUERTAS

Centronics en paralelo y cinco ranuras

LENGUAJES DISPONIBLES

BASIC, más una selección disponible bajo PC-DOS, que incluye COBOL, FORTRAN, etc.

TECLADO

79 teclas tipo máquina de escribir

DOCUMENTACION

Responde al estándar normal que se puede esperar de IBM y de los proveedores de software que ha escogido

VENTAJAS

El IBM PC básico se puede utilizar de forma bastante exitosa y después ampliarlo y mejorarlo convirtiéndolo en uno de los microordenadores más potentes de cuantos existen

DESVENTAJAS

Es muy caro si se compara con las muchas imitaciones del PC que aparecieron tras su lanzamiento. El servicio y el software son asimismo más caros que los de máquinas más sencillas



Teclado
Como se podría esperar de uno de los más grandes fabricantes de máquinas de escribir, terminales de ordenador y otros dispositivos activados por teclado, el teclado del IBM PC es virtualmente impecable. Está separado de la unidad procesadora, para que el usuario pueda colocarlo en la posición que desee, su perfil es sumamente plano y de inclinación ajustable, y utiliza teclas esculpidas dispuestas en cinco filas

Chris Stevens

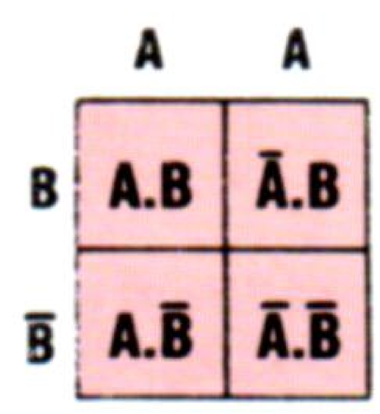
El Peanut
El PC Junior de IBM, al que durante la fase de desarrollo se aludía en clave como el *Peanut* (cacahuete), es una versión notablemente rebajada de la máquina más grande. Quizá la innovación más interesante fuera el que se utilizara un enlace infrarrojo entre teclado y procesador, en lugar del cable habitual. El PC Jr está equipado con dos ranuras para cartucho

Lectura de diagramas

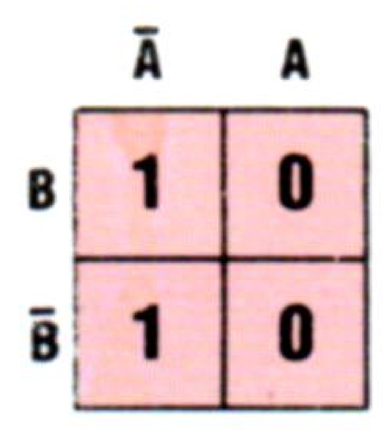
Una valiosa ayuda para la simplificación de los circuitos lógicos son los diagramas de Karnaugh. Las más complejas expresiones algebraicas se vuelven diáfanas con este método

Los *diagramas de Karnaugh* (también llamados *diagramas k*) son en realidad ampliaciones de los diagramas de Venn que ya conocemos (véase p. 526). Nos permiten representar las expresiones lógicas de forma gráfica. Un diagrama k asume formas ligeramente distintas según el número de letras (o variables) diferentes que haya en la expresión a simplificar, y es de mayor utilidad para las expresiones que contengan dos, tres o cuatro variables.

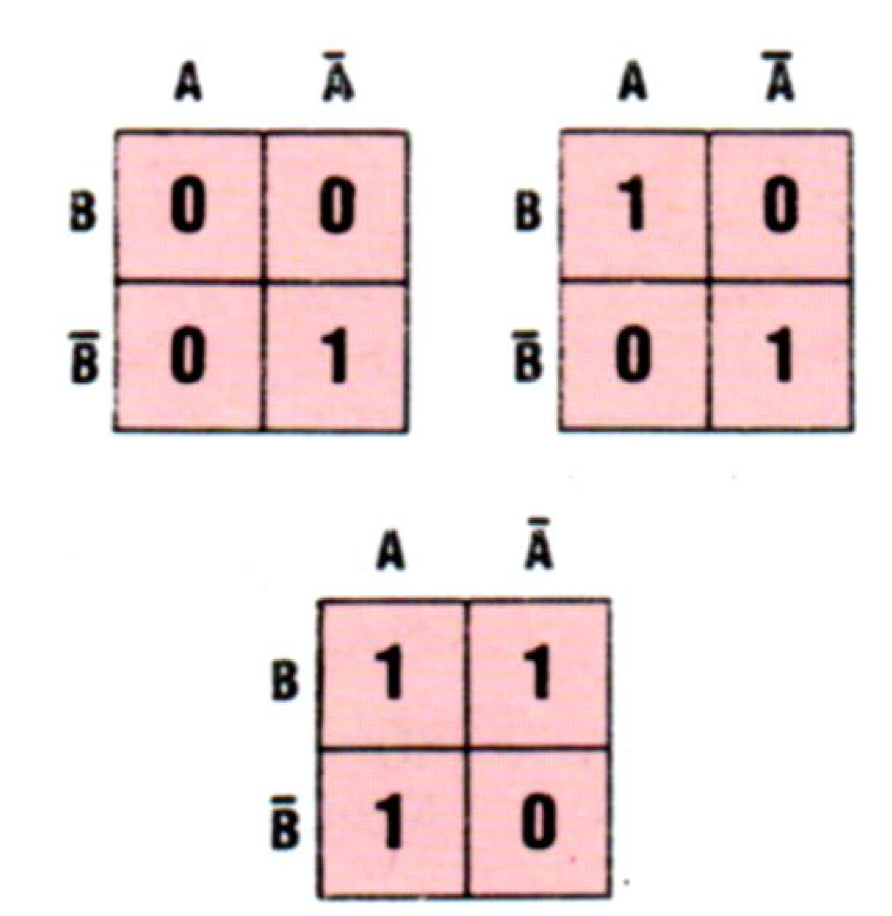
Dos variables: Cada uno de los cuadros de un diagrama k de dos variables (habrá $2^2 = 4$ cuadrados) representa una función AND, tal como refleja este diagrama:



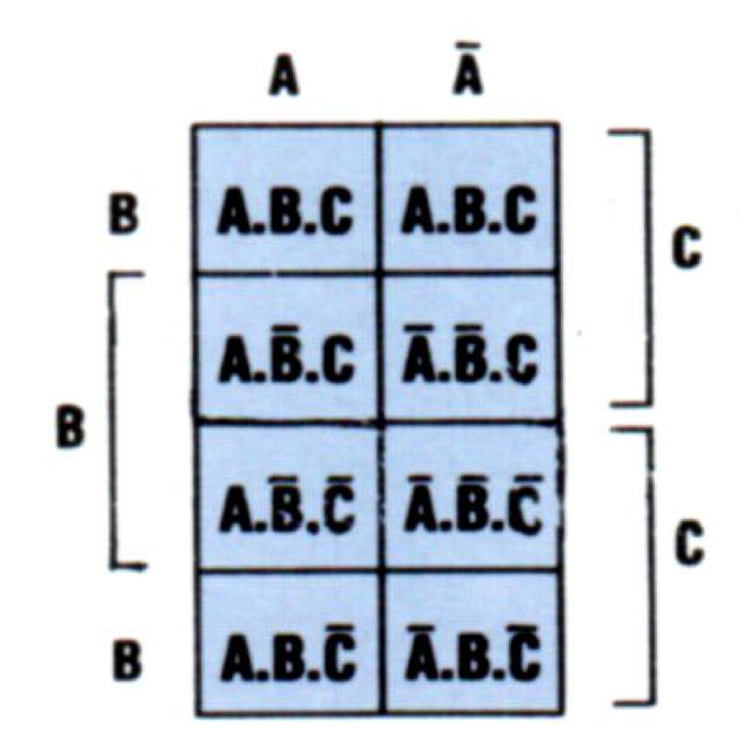
Para representar la expresión $AB + A\bar{B}$ como un diagrama k, colocamos unos en los cuadrados pertinentes:



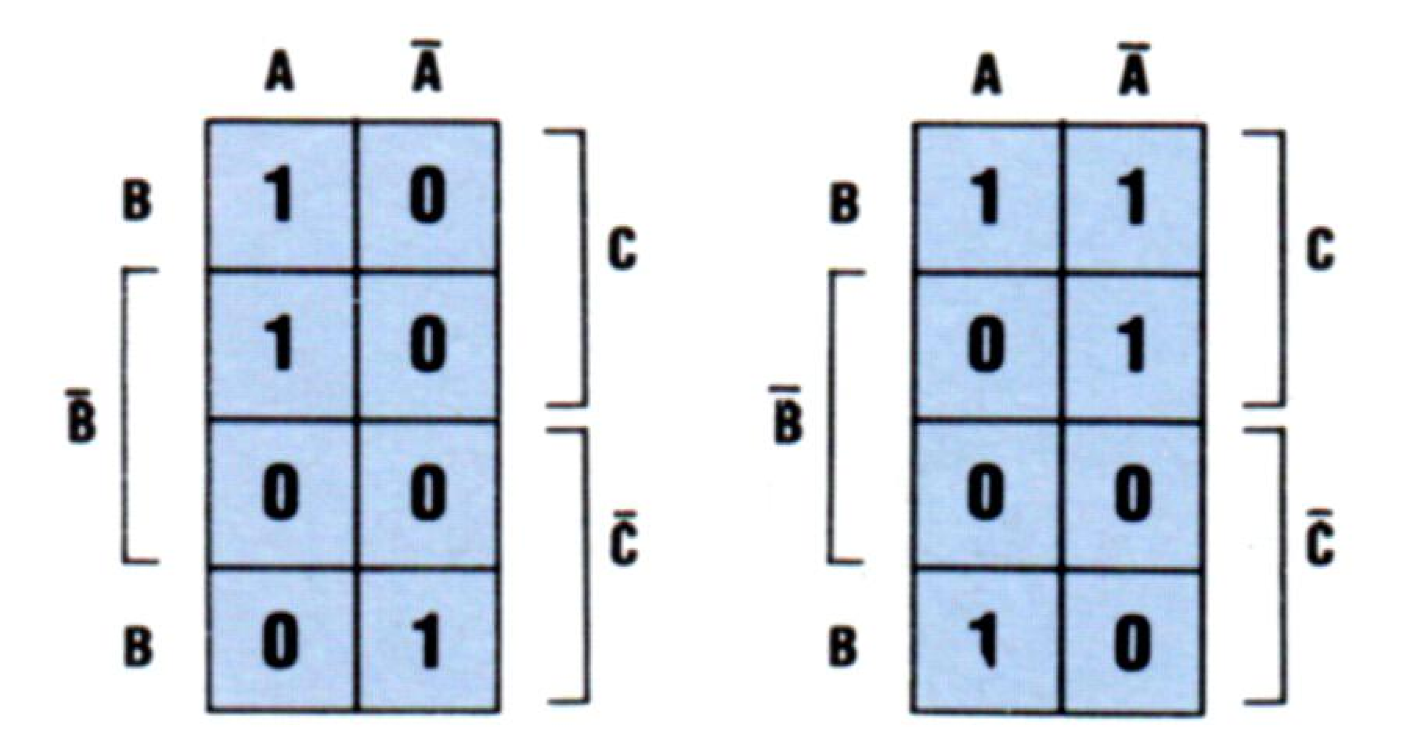
Aquí tenemos otros tres ejemplos, que representan respectivamente las expresiones $\bar{A}\bar{B}$, $AB + \bar{A}\bar{B}$ y $AB + \bar{A}B + A\bar{B}$:



Tres variables: En este caso, el número de cuadrados se multiplicaba por dos ($2^3 = 8$ cuadrados). El diagrama k básico de tres variables es:



He aquí dos expresiones, $AC + \bar{A}B\bar{C}$ y $AB + \bar{A}C$, representadas como diagramas k:

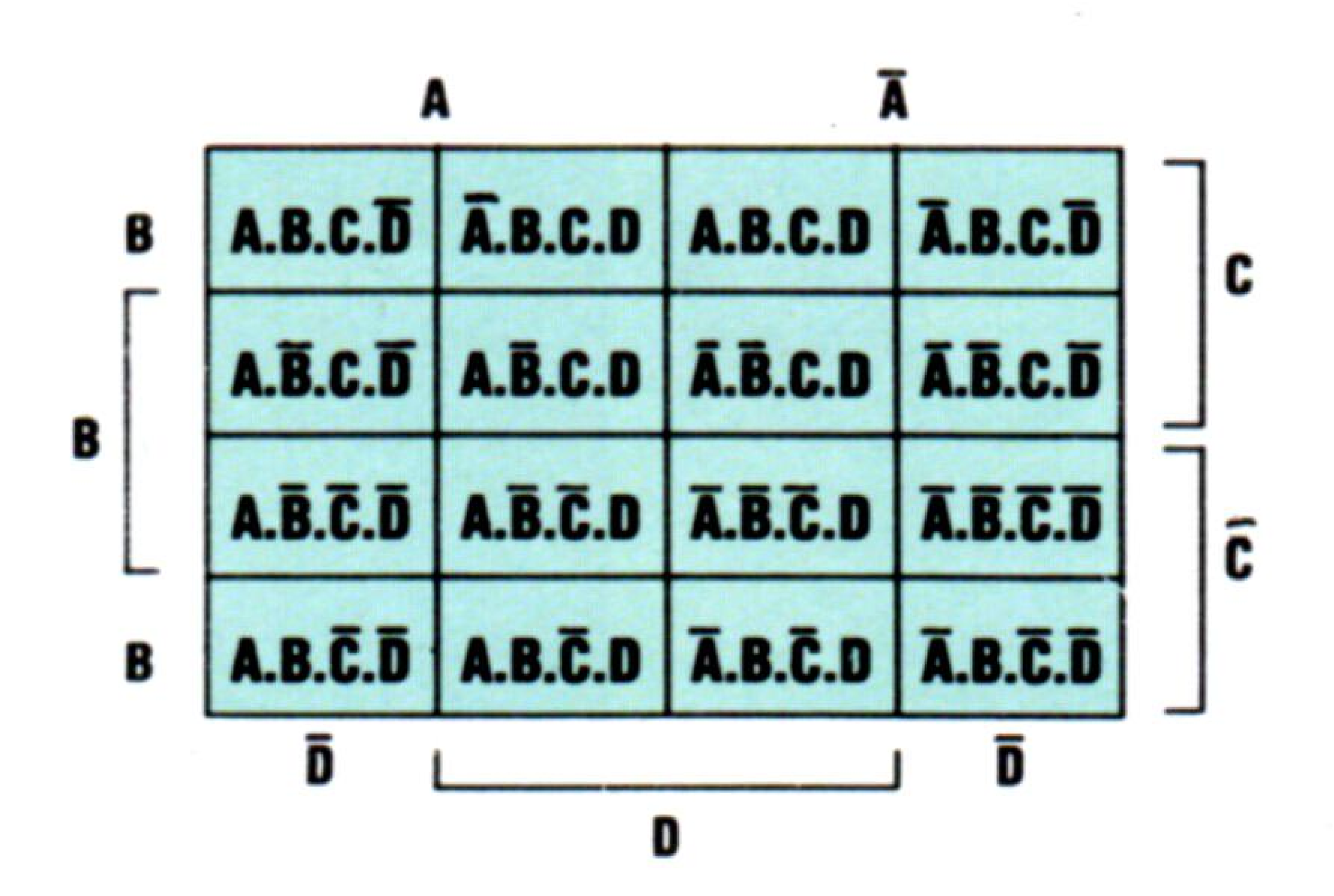


$$ABC + A\bar{B}C + \bar{A}B\bar{C}$$
$$= AC(B + \bar{B}) + \bar{A}B\bar{C}$$
$$= AC + \bar{A}B\bar{C}$$

$$ABC + AB\bar{C} + \bar{A}BC + \bar{A}\bar{B}C$$
$$= AB(C + \bar{C}) + \bar{A}C(B + \bar{B})$$
$$= AB + \bar{A}C$$

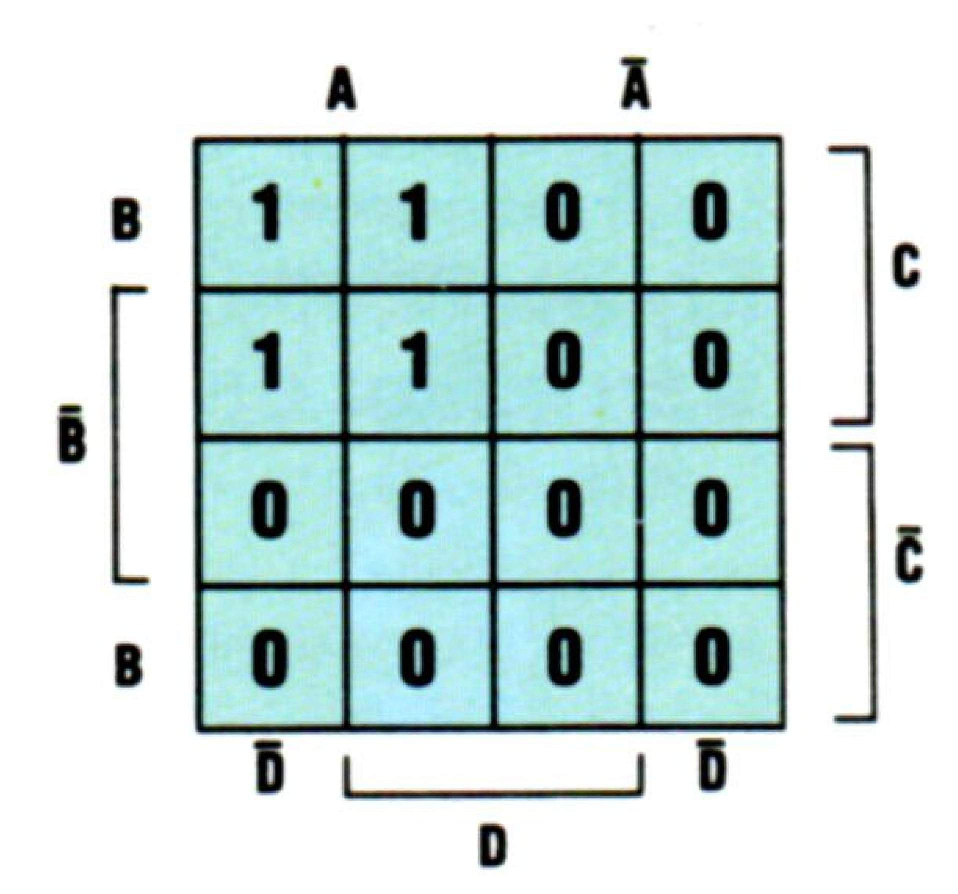
Observe que ambas expresiones se han simplificado utilizando la ley booleana según la cual un conjunto A relacionado mediante OR con su negativo ($\bar{A}$) da como resultado 1 (el conjunto universal o identidad).

Cuatro variables: Cuando comenzamos a tratar con cuatro variables, los mapas empiezan a hacerse más complicados (tienen $2^4 = 16$ cuadrados), pero no obstante son bastante fáciles de interpretar de acuerdo a la cuadrícula básica:



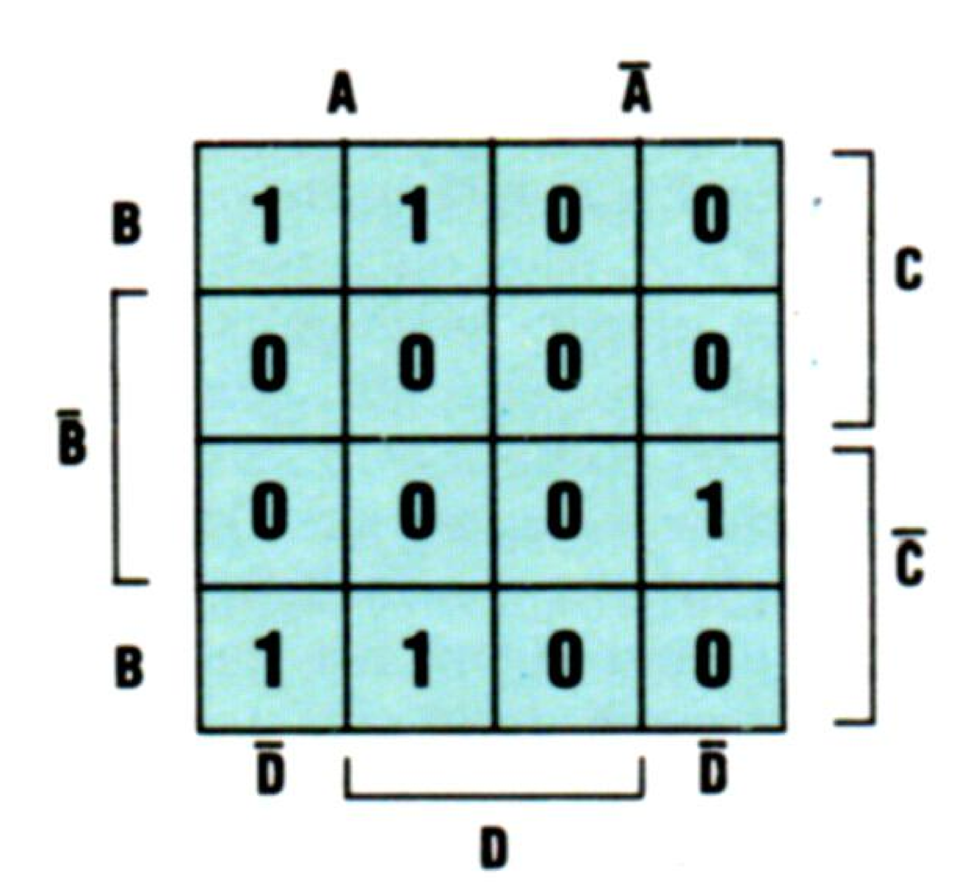


He aquí un diagrama k acompañado de una simplificación:



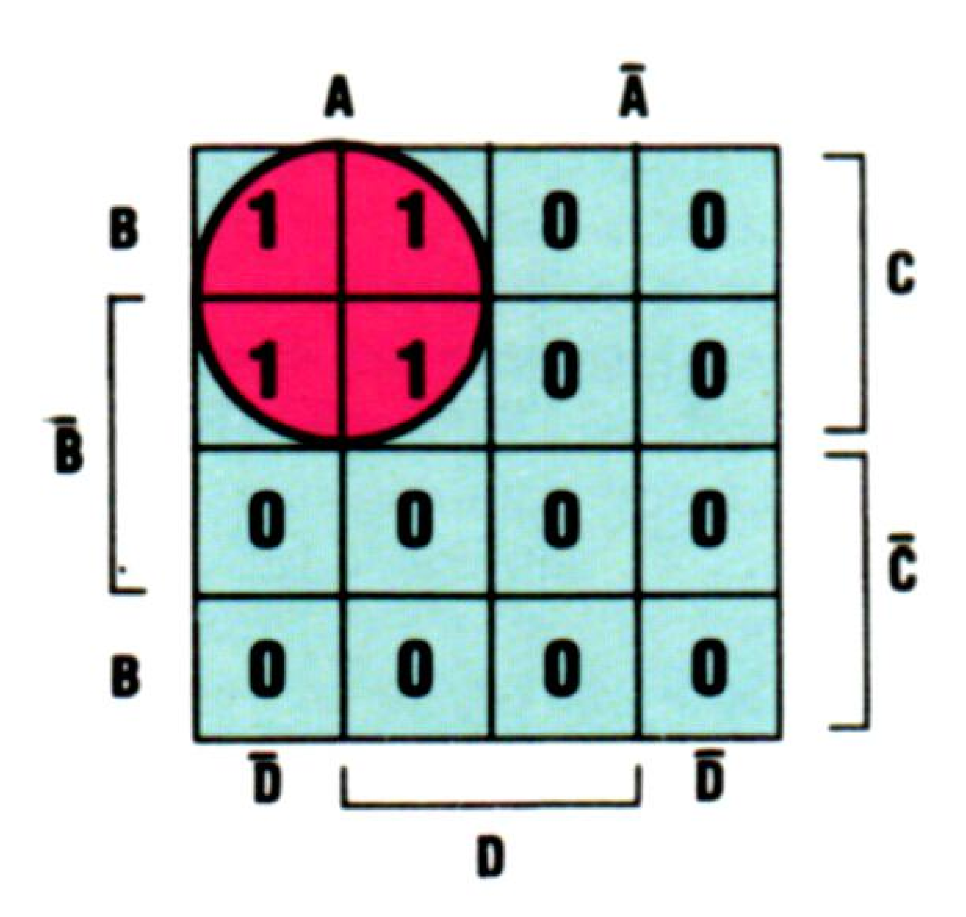
$$ABC\bar{D} + ABCD + A\bar{B}C\bar{D} + A\bar{B}CD$$
$$= ABC(\bar{D} + D) + A\bar{B}C(\bar{D} + D)$$
$$= ABC + A\bar{B}C$$
$$= AC(B + \bar{B})$$
$$= AC$$

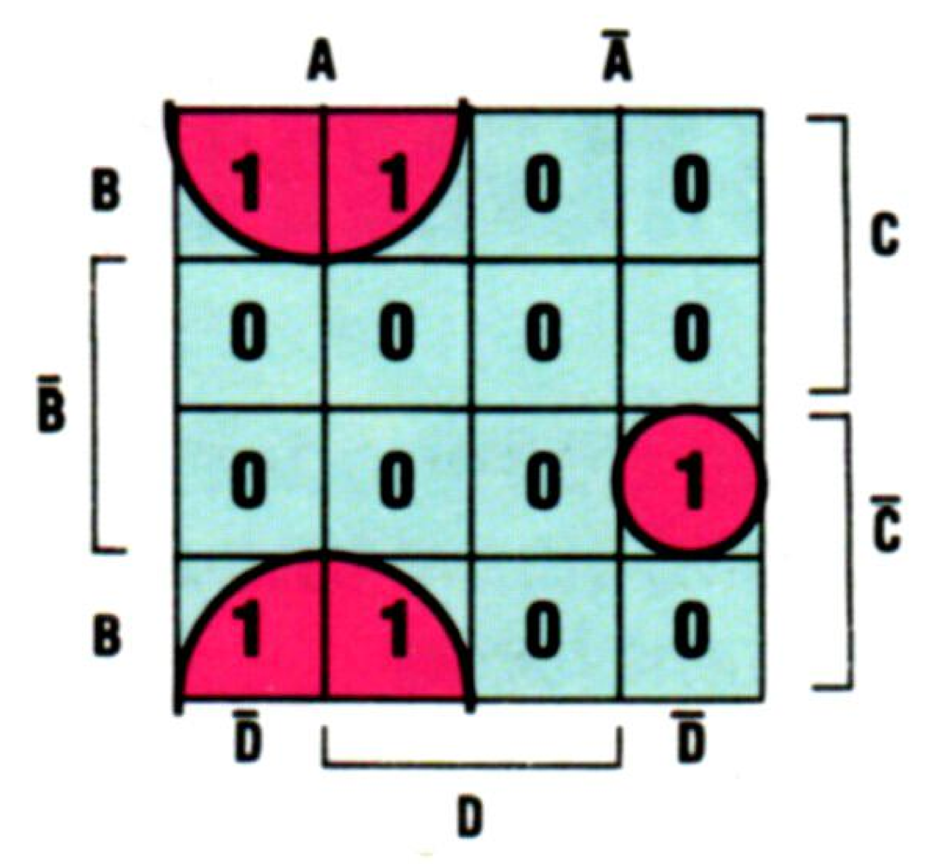
He aquí otro ejemplo:



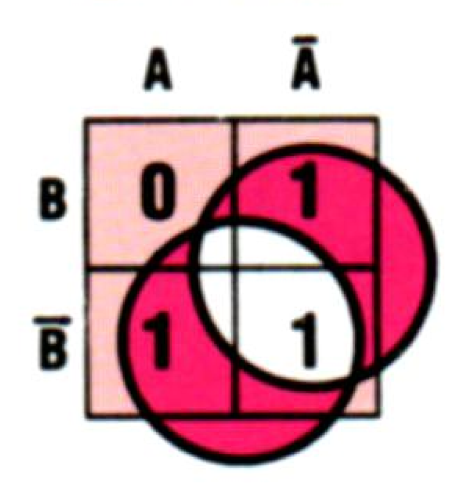
$$ABC\bar{D} + ABCD + \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + AB\bar{C}D$$
$$= ABC(\bar{D} + D) + \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}(\bar{D} + D)$$
$$= ABC + \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}$$
$$= AB(C + \bar{C}) + \bar{A}\bar{B}\bar{C}\bar{D}$$
$$= AB + \bar{A}\bar{B}\bar{C}D$$

Si analizamos con más detenimiento la disposición de los unos en estos dos diagramas k, podremos descubrir cierta regularidad. En el primer ejemplo, todos los cuadrados con AC en sus expresiones poseen un 1. En el segundo ejemplo, lo mismo sucede con todos los cuadrados AB. Esto sugiere que una forma más sencilla de simplificar expresiones booleanas consiste simplemente en inspeccionar un diagrama k. Consideremos los siguientes:



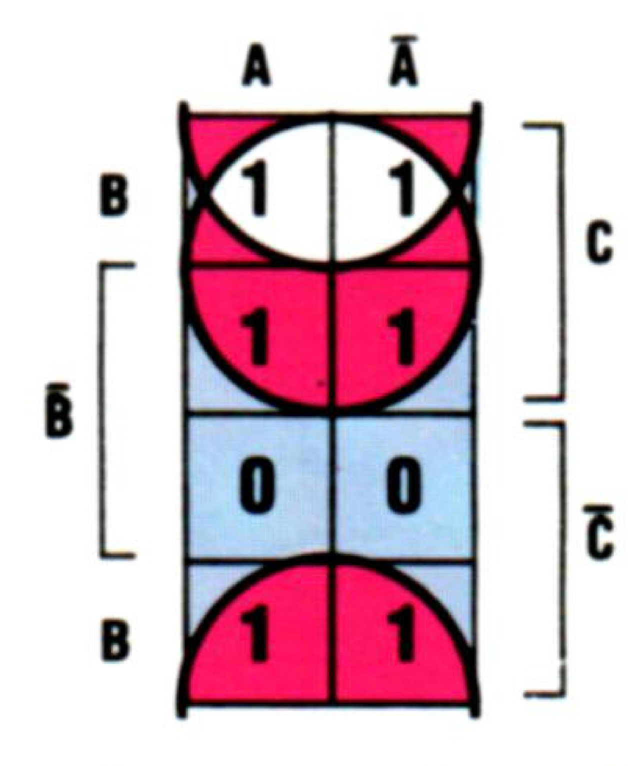


Con un poco de práctica, se pueden detectar grupos de dos, cuatro u ocho unos para formar términos más simples. Por ejemplo, consideremos esta expresión: $\bar{A}B + A\bar{B} + \bar{A}\bar{B}$.



Utilizando un diagrama k de dos variables, podemos detectar dos grupos de unos. Un grupo representa todos los casos NO(B) y el otro representa todos los casos NO(A), de manera que podemos simplificar la expresión a $\bar{A} + \bar{B}$. Esta expresión se puede volver a simplificar, mediante la ley de Morgan: $\overline{A.B}$. ¿Se puede llegar a la conclusión de forma más directa inspeccionando el diagrama k?

Un ejemplo más difícil lo ofrece esta expresión de tres variables:
$AB\bar{C} + A\bar{B}C + \bar{A}BC + AB\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C$



El grupo de cuatro unos en la parte superior del diagrama k representa todos los posibles casos en los que C es verdadera. Las filas superior e inferior del mapa representan todos los posibles casos en los que B es verdadera. Por consiguiente, la expresión simplificada es: B + C.

En el próximo capítulo del curso proseguiremos con nuestra investigación acerca de la utilización de los diagramas k para simplificar expresiones booleanas que comprenden cuatro variables. Luego le mostraremos cómo se utilizan estos diagramas en el proceso del diseño de circuitos. Y ello nos permitirá unificar todos los aspectos del curso que hemos analizado hasta el momento.

Ejercicio 4:
Dibuje diagramas k de tres variables para simplificar las siguientes expresiones booleanas:

a) $\bar{A}B.C + \bar{A}.\bar{B}.C + \bar{B}.\bar{C} + \bar{A}.B.\bar{C}$
b) $A.\bar{B}.\bar{C} + \bar{A}.B.\bar{C} + A.B.\bar{C}$

Sencillo y lógico

El BASIC de Commodore no es una versión muy avanzada, pero tiene una lógica y una simplicidad notables y su editor de pantalla es de los mejores

Todas las máquinas CBM admiten variables con nombres largos, pero el intérprete sólo explora los dos primeros caracteres del nombre, de modo que, por ejemplo, tanto FUSION como FUERZA son admisibles, pero equivalentes. Por consiguiente, la salida de este fragmento:

```
100 FUSION = 17: FUERZA = 2*FUSION
200 PRINT FUSION, FUERZA
```

es:

```
34        34
```

Esto se aplica a todos los tipos de variables: de coma flotante (p. ej., NUMERO), de enteros (p. ej., NUMERO%), en serie (p. ej., NUMERO$) y de matriz (p. ej., NUMERO$(62,47)). Los tipos de variables son convencionales, pero en el Vic-20 las variables de enteros son inutilizables porque la máquina no admite aritmética de enteros; el tipo entero se conservó simplemente para mantener la compatibilidad con otras máquinas Commodore que admiten aritmética de enteros.

Una negativa consecuencia de estas reglas sobre los nombres de las variables es que un nombre que pareciera válido podría ser invalidado porque sus dos primeras letras conformaran una palabra reservada (START, p. ej., es equivalente a ST como nombre de variable, y ST es una palabra reservada).

Las variables de matriz pueden tener hasta 255 dimensiones y su extensión sólo está limitada por la cantidad de RAM disponible. El primer elemento de cualquier matriz es el elemento(0), de modo que DIM EX(6) crea una matriz de siete elementos: EX(0), EX(1), EX(2),..., EX(6). Aquí la sentencia DIM es innecesaria, porque si el intérprete se encuentra con una variable de matriz de dimensión simple para la cual no se haya ejecutado ninguna sentencia DIM, se sobreentiende, por omisión, una dimensión 10; si el subíndice de dicha matriz es mayor que 10, entonces se producirá un error BAD SUBSCRIPT (subíndice malo). Ésta es una facilidad arbitraria que no favorece una buena práctica de programación: el intérprete debe reacomodar la memoria cada vez que se encuentra con una sentencia DIM (o la primera referencia a una matriz no DIMensionada), de modo que todas las matrices se deben DIMensionar al mismo tiempo al comienzo del programa, antes de que se emplee ninguna variable simple. Si esto no se hace no sucederá nada grave, pero incidirá ligeramente en la velocidad de ejecución.

Debido a la forma en que trabaja la mayoría de los intérpretes de BASIC, la ejecución de los programas se puede acelerar inicializando las variables del programa utilizadas más comúnmente por su orden de importancia; esto se puede hacer con sentencias de asignación o con la DIM. Una línea como:

```
10 DIM A$(10,24),K,L,MARCADOR
```

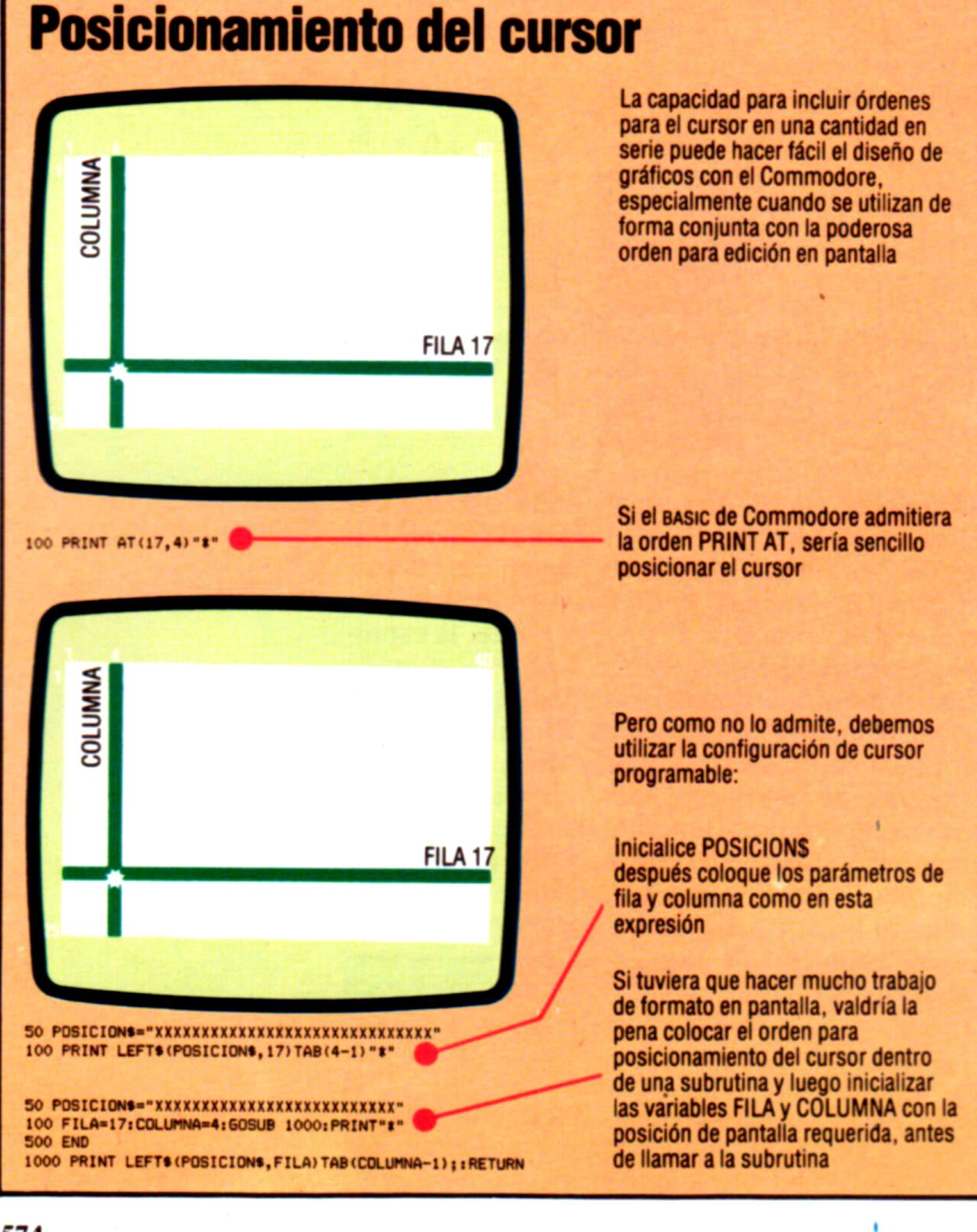
no tendrá ninguna consecuencia obvia, pero se trata de una forma rápida de colocar las variables K,L y MARCADOR arriba de la tabla de símbolos, haciéndolas, por lo tanto, más accesibles al intérprete y aumentando la velocidad de ejecución.

Una mirada a la lista de las palabras clave en un manual para el usuario CBM (del que hablaremos más adelante) revela unas pocas omisiones, y algunos añadidos, al juego Microsoft completo. Tal vez la omisión más importante sea INKEY$ y las adiciones más significativas TIME$ y STATUS.

INKEY$, la función para exploración del teclado, se sustituye por la sentencia GET. Al igual que INKEY$, hace que se explore el primer carácter del buffer del teclado y devuelve su valor ASCII. GET se suele usar más a menudo en sentencias como:

```
150 GET GT$:IF GT$ = "" THEN 150
```

Posicionamiento del cursor



La capacidad para incluir órdenes para el cursor en una cantidad en serie puede hacer fácil el diseño de gráficos con el Commodore, especialmente cuando se utilizan de forma conjunta con la poderosa orden para edición en pantalla

Si el BASIC de Commodore admitiera la orden PRINT AT, sería sencillo posicionar el cursor

Pero como no lo admite, debemos utilizar la configuración de cursor programable:

Inicialice POSICION$ después coloque los parámetros de fila y columna como en esta expresión

Si tuviera que hacer mucho trabajo de formato en pantalla, valdría la pena colocar el orden para posicionamiento del cursor dentro de una subrutina y luego inicializar las variables FILA y COLUMNA con la posición de pantalla requerida, antes de llamar a la subrutina

Liz Dixon



que tendrá el efecto de detener la ejecución del programa hasta que se pulse una tecla, en cuyo caso GT$ contendrá el carácter correspondiente a la tecla pulsada. Puede que no siempre sea éste el caso, porque GET explora el buffer del teclado y no el teclado propiamente dicho, de modo que si cuando se efectúa GET el buffer contiene algunos caracteres, entonces la ejecución del programa no esperará a que el usuario pulse una tecla. Ello se puede demostrar mediante este programa:

```
50 FOR K = 1 TO 100:PRINT K:NEXT K
60 PRINT "PULSE CUALQUIER TECLA"
150 GET GT$:IF GT$ = "" THEN 150
200 PRINT "USTED HA PULSADO LA TECLA";GT$
```

Si ejecuta este programa verá que en la pantalla aparecen los números del 1 al 100, seguidos del mensaje para la entrada, y después nada hasta que usted pulsa una tecla. Sin embargo, si pulsa una tecla mientras se están imprimiendo los números, entonces esa pulsación penetrará en el buffer y allí la encontrará la sentencia GET, de modo que no se producirá ninguna pausa en la ejecución del programa. Esto puede ser fastidioso y posiblemente desastroso, por ejemplo, en los juegos, donde se producen muchas pulsaciones frenéticas de teclas. La solución consiste en restaurar el buffer justo antes de que se ejecute la sentencia GET, mediante la inserción en el programa de:

```
149 POKE KBPTR,0
```

donde KBPTR es la dirección del contador de cola de teclado (198 en el Commodore 64).

Normalmente GET no genera un cursor centelleante en la pantalla, pero POKE FLASH,0 —donde FLASH es la dirección de la bandera de activación del centelleo de cursor (204 en el Commodore 64)— proporcionará uno.

TIME$ (que se suele abreviar TI$) es el reloj del sistema; en el momento del encendido se inicializa a "000000" y a partir de entonces marca el tiempo en horas, minutos y segundos hasta "235959" (23 horas 59 minutos 59 segundos desde la inicialización), momento en que se restaura a "000000". Está a disposición del usuario al igual que cualquier otra variable, y se puede establecer en cualquier hora legal, tal como: TI$ = "000000" o como TI$ = "084503".

Relacionado con TI$ está TI, su equivalente numérico. TI marca la hora desde la inicialización expresada en sesentavos de segundo (denominados *jiffys*), de modo que su valor oscila entre 0 y 5183999 (60*60*60*24-1). TI depende de TI$ y no se puede inicializar por sí mismo; sólo se inicializa TI$.

STATUS (que se abrevia ST) es una variable definida por el sistema. Cuando se detecta un error en un dispositivo de entrada-salida, el valor de ST será un número que indique el tipo de error detectado. Éste se escribiría así:

```
330 IF ST > 0 THEN GOSUB 30000:REM ERROR EN
    MANIPULACION DE ARCHIVO E/S


30000 REM MENSAJES DE ERROR
30100 IF ST = 16 PRINT "ERROR IRRECUPERABLE
      DE LECTURA"
```

CMD es un miembro muy útil del juego de instrucciones Commodore. Tiene el efecto de desviar la salida de la pantalla a un canal de salida seleccionado. Posee muchas aplicaciones útiles, como por ejemplo:

```
OPEN 4,4:CMD 4:LIST
```

Esto LISTa el programa corriente a la impresora en vez de a la pantalla; cuando el listado esté completo se ha de ejecutar:

```
PRINT#4:CLOSE 4
```

CMD se puede utilizar en un programa para copiar la pantalla en la impresora. Supongamos que en su programa GOSUB 3000 hace que se imprima (PRINT) en la pantalla (en vez de que se coloque —POKE— en la memoria de pantalla) un mensaje o algunos datos. Para poder copiar esa visualización en la impresora, digite:

```
OPEN4,4:CMD 4:GOSUB3000:PRINT#4:CLOSE4
```

Aunque se acepta un signo de interrogación (?) como abreviatura de la palabra clave PRINT, no se puede abreviar PRINT# por ?#; para hacer esto debe utilizar pR (p seguida de r con tecla de cambio).

Las abreviaturas de las palabras clave Commodore son tan antiguas como las propias máquinas Commodore, pero oyendo a los usuarios de ordenadores Spectrum se podría pensar que la idea la concibió Sinclair. Casi todas las palabras clave se pueden abreviar con su letra inicial seguida de la segunda letra mayúscula. Cuando dos palabras clave o más poseen las mismas dos primeras letras, entonces la abreviatura consistirá en las dos primeras letras más la tercera cambiada: READ, RESTORE y RETURN, por ejemplo, se abrevian rE, reS y reT.

El editor de pantalla, y el sólido a la vez que cómodo sistema operativo sobre el que se sustenta, constituye la configuración individual más significativa de las máquinas Commodore. Se viene utilizando desde el lanzamiento del PET y sigue siendo uno de los mejores editores de pantalla para micros que existen. Para editar una línea de programa, por ejemplo, LISTe la línea, desplazar el cursor directamente hasta cualquier punto de la línea, editar el texto y pulsar RETURN. No importa dónde esté el texto en la pantalla ni dónde esté el cursor en la línea: cuando pulsa RETURN el texto de la línea en pantalla que contiene el cursor entra en el sistema como si usted lo hubiera digitado.

Una sutileza de este sistema de edición es la facilidad que otorga para copiar. Supongamos que tiene que dar entrada a estas dos líneas:

```
100 IF  INT(NUMERO/INDICE—TASA) = 5  THEN
    3000
200 IF INT(NUMERO/INDICE—TASA) = 7 THEN
    3800
```

Sólo necesita digitar la primera línea y pulsar RETURN; luego, con ese texto en la pantalla, desplaza el cursor hacia arriba, cambia el número de línea 100 por 200, cambia 5 por 7, cambia 3000 por 3800 y vuelve a pulsar RETURN. En la memoria la línea 100 permanece tal como estaba y su texto editado en la pantalla se convierte en la línea 200. Este proceso se puede repetir tantas veces como lo desee. Ésta no es más que una de las fascinantes facilidades que uno puede obtener con el editor, pero demuestra lo fácil y directa que resulta su utilización.

Superestafa

En 1971 Jerry Schneider estafó 1 000 000 de dólares en equipos a la empresa Los Angeles Pacific Telephone and Telegraph. Recuperando manuales y equipos de los depósitos de desperdicios de esa compañía, se hizo pasar por periodista y consiguió los códigos de acceso para el ordenador IBM 360 de la empresa. Schneider hacía pedidos discretos y revendía la mercancía. Tras pasar 40 días en prisión, se colocó como consultor en seguridad de ordenadores.

Líneas de Assembler

Vamos a revisar los principales recursos utilizados para tratar con la memoria y veremos, además, las diferencias entre los programas para el 6502 y el Z80

Cuando se ejecuta (RUN) un programa, lo primero que hace el sistema operativo es inspeccionar los indicadores de comienzo de texto en BASIC con el fin de determinar en qué lugar de la memoria reside el programa a ejecutar. Para hacer esto, sin embargo, el sistema operativo tiene que almacenar las direcciones de los indicadores. Entonces ¿por qué el OS no almacena sencillamente las direcciones indicadas?

La razón principal es la flexibilidad. El sistema operativo, como recordará (véase p. 84), es un programa permanente residente en la ROM, y cualquier dato que contenga (como las direcciones de memoria) es igualmente permanente. Supongamos que durante un tiempo determinado se fueron lanzando distintas versiones de un ordenador y que, mientras en la versión 1 fue conveniente tener el comienzo del texto en BASIC en el byte 2048, en la versión 2 se hizo necesario cambiarlo de lugar y colocarlo en el byte 4096. Ello significará que la última máquina no será capaz de utilizar el sistema operativo de la versión anterior, debido a la diferente posición del área para textos en BASIC. Además, habría que crear nuevas ROM para cada nueva versión de la máquina, lo que resulta costoso; y bien podría ser que el software escrito para una versión no se pudiera emplear con la otra. Por el contrario, si las ROM del sistema operativo sólo contienen las direcciones de los indicadores, entonces se pueden utilizar estas mismas para todas las versiones de la máquina y de un modelo a otro sólo se necesitará cambiar el contenido de los mismos. La posición de los indicadores propiamente dichos puede permanecer constante, porque el sistema operativo requiere un bloque de memoria relativamente pequeño para espacio de trabajo y almacenamiento de datos (por lo común, alrededor de 1 000 bytes). Fijar la posición de este bloque (habitualmente las primeras cuatro páginas de la memoria) y diseñar o rediseñar el sistema alrededor del mismo no supone un gran impedimento para el equipo de diseño. Sin embargo, tener fija, pongamos por caso, la posición del área para textos en BASIC (un bloque de entre 3 000 y 40 000 bytes) sí supone una gran limitación.

La práctica estándar

Es práctica generalizada almacenar las direcciones de los indicadores en la forma que se conoce como *lo-hi* (apócope de *low-high*: "bajo-alto", en inglés). Si el byte 43 y el byte 44, por ejemplo, han de señalar la dirección 7671 (página 29, desplazamiento 247), entonces el byte 43 contendrá 247 (el desplazamiento o byte *lo*—bajo— de la dirección), mientras que el byte 44 contendrá 29 (la página o byte *hi*—alto— de la dirección). Esto puede confundir al principio, pero resulta conveniente para el microprocesador. También es lógico que el byte *lo* de la dirección se almacene en el byte *lo* del indicador, y el byte *hi* de la dirección en el *hi* del indicador.

Si repetimos el ejemplo anterior utilizando números hexas en vez de números decimales, podremos apreciar la gran ventaja del sistema hexadecimal (de ahora en adelante, las direcciones y otros números siempre se escribirán en hexa precedidos por el signo $). Los bytes señaladores son $2B y $2C, y la dirección a la que señalan es $1DF7. Por consiguiente, $2B contiene F7 (el byte *lo* de la dirección), mientras que $2C contiene $1D (el byte *hi* de la dirección). Observe que cuando la dirección está en hexa los dos dígitos hexas a la derecha son el byte *lo*, y los dos dígitos a la izquierda son el byte *hi*, lo que tiene mucho más sentido que utilizar números decimales.

Es importante destacar que el BBC y el Spectrum se desvían de esta norma, ya que almacenan los números de línea del programa como números de dos bytes en forma *hi-lo* (alto-bajo) en lugar de *lo-hi* (bajo-alto). Cierto que son parámetros del programa en vez de direcciones de bytes, pero así y todo están al reves de la convención habitual.

Otra práctica corriente en el direccionamiento de memoria es la de denominar los indicadores mediante la dirección del byte *lo* solamente a pesar de que son cantidades de dos bytes. Podríamos decir, por ejemplo, que en el Commodore 64 el byte 43 señala el comienzo del texto en BASIC. En este caso se sobreentiende, sin embargo, que el byte 43 y el byte 44 son conjuntamente los indicadores.

Otro aspecto a considerar son los distintivos, o *tokens* (véase p. 556). Tienen una significación doble para los programadores en lenguaje máquina: representan órdenes en inglés de múltiples caracteres (como PRINT o RESTORE) mediante códigos numéricos de un único byte; y utilizan además desplazamientos. En BASIC, una orden es una palabra, pero para el sistema operativo ejecutarla no es una sola operación. La orden PRINT, por ejemplo, exige que se hallen en la memoria o evalúen los datos a imprimir, para después enviarlos carácter por carácter en código ASCII. Estas diversas tareas se llevan a cabo mediante una subrutina del programa del intérprete de BASIC. Cuando el intérprete encuentra en una línea de programa el distintivo PRINT, toma el valor de ese distintivo para localizar y ejecutar la correspondiente subrutina.

Supongamos que en nuestra versión de BASIC sólo hay tres órdenes: INPUT, PRINT y STOP; y éstas tienen asignados los distintivos $80, $81 y $82, respectivamente. Además, vamos a suponer que las subrutinas del intérprete que ejecutan estas órde-



Paso a paso

Este recuadro muestra cómo se traduce y se ejecuta una línea de programación en BASIC y una instrucción en lenguaje máquina

El sistema operativo transmite los datos de la línea en la forma habitual, sustituyendo las sentencias del BASIC por los correspondientes distintivos

Aquí se digita RUN

El intérprete de BASIC busca en la línea los distintivos y los datos relacionados con los mismos, utilizando el valor del distintivo para localizar la subrutina de manipulación del sistema operativo apropiada

El ensamblador traduce las expresiones mnemotécnicas del lenguaje Assembler en opcodes de un byte y almacena el operando de 2 bytes en forma *lo-hi*

Cuando se ejecuta la instrucción, el microprocesador decodifica el opcode en códigos de longitud y de operación, para así tratar como operando al número correcto de bytes que siguen al opcode

Kevin Jones

nes empiezan en los bytes $D010, $EA97 y $EC00 respectivamente, y que estas tres direcciones están almacenadas en forma *lo-hi* en los seis bytes a partir de $FA00 y hasta $FA05, que nos proporcionan una tabla de tres indicadores de dos bytes. Pues bien, cuando nuestro intérprete imaginario encuentra un distintivo ($81, p. ej.) procede a restarle $80, multiplica el resultado por dos y lo suma a $FA00. El resultado final en este caso es $FA02, que es el byte *lo* del indicador para la subrutina PRINT. Si se hubiera encontrado con un distintivo diferente a $81, entonces el algoritmo descrito habría devuelto la dirección del indicador para la subrutina correspondiente. De esta manera, la orden PRINT se reemplaza por un distintivo, $81, que es un desplazamiento de una tabla de indicadores que dirige al intérprete hasta la parte en cuestión de su propio programa.

Tenemos aquí una medida de la "distancia" entre el BASIC, un lenguaje denominado de alto nivel, y el lenguaje máquina, o lenguaje de bajo nivel. A nosotros el BASIC nos parece comprensible porque utiliza palabras de un código tan asimilable como es la lengua inglesa, la lógica algebraica y los números y series. Cuando sustituimos las palabras por distintivos y el resto por códigos ASCII, se parece ya a algo que el microprocesador puede manipular.

Por último, estudiemos la noción de *contexto*. En el área para textos en BASIC hemos visto la extendida utilización de los códigos: códigos ASCII para representar caracteres y números, distintivos para representar órdenes y (en el Spectrum) códigos binarios especiales para representar datos numéricos. Todos estos códigos se reducen a números binarios en la escala entre 00000000 y 11111111 (de $00 a $FF, de 0 a 255 en decimal) contenidos en bytes individuales de memoria e interpretados de acuerdo a su contexto. Dentro del área para textos en el BASIC del Commodore 64, la línea de programa:

```
200 rem*******left$*********
```

podría tener tres bytes conteniendo el número decimal 200: uno para el byte *lo* de la dirección de enlace, otro para el byte *lo* del número de línea y el tercero en la representación del distintivo de "left$". Cada byte tiene el mismo aspecto que los otros, y sin embargo significa algo diferente. Sólo sus expectativas le dirán a usted cómo interpretar ese valor en distintas situaciones.

Y aquí es donde realmente abordamos el inicio de estas lecciones de lenguaje máquina. Dijimos entonces que todo lo que hay almacenado en un ordenador está en algún tipo de lenguaje máquina. Parte de ello era familiar (como los códigos ASCII), parte desconocido (como los distintivos) y el resto quedaba sin explicar (como los programas en lenguaje máquina). Toca, pues, explicar estos mismos programas en lenguaje máquina.

Códigos de operación (opcodes)

Los programas en lenguaje máquina se reducen a unas cuantas secuencias de bytes situados en algún lugar de la memoria, que representan a su vez una mezcla de instrucciones para el microprocesador y datos sobre los cuales ha de operar éste. De modo semejante a lo que sucede con los demás bytes de la memoria, sólo el contexto puede separar los bytes de datos de los bytes de instrucciones, por lo que primero debemos considerar el formato de las instrucciones de un programa en lenguaje máquina.

Una instrucción en lenguaje máquina empieza con un código que indica la operación a realizar. Se denomina *código de operación* (abreviatura inglesa: *opcode* u *opc*) y puede tener uno o dos bytes de longitud. El opcode puede ser una instrucción cuya ejecución no necesite datos, pero las más de las veces va seguido de uno o dos bytes de datos. Un byte individual de datos puede ser una constante numérica o un código ASCII, mientras que dos bytes de datos a continuación de un opcode siempre son una dirección (almacenada en la forma byte *lo*-byte *hi*). Con la descripción anterior surgen de inmediato las diferencias existentes entre los microprocesadores: el BBC Micro utiliza un MOS Tech 6502A, el Commodore 64, un MOS Tech 6510 (muy similar al 6502A, de modo que en el futuro hablaremos generalmente sólo del 6502), y el Spectrum posee el Zilog Z80A. MOS Tech y Zilog produjeron sus microprocesadores aproximadamente al mismo tiempo (a comienzos de los años setenta), después que Intel lanzara el primer microprocesador, en 1971. Por consiguiente, tanto el 6502 como el Z80 comparten un mismo criterio de diseño, pero difieren sustancialmente en los detalles. En particular, los códigos de lenguaje máquina del Z80 son completamente distintos de los códigos de lenguaje máquina del 6502. Así, por ejemplo, los opcodes del 6502 siempre son de un byte de largo y pueden ir seguidos por uno o dos bytes de datos o por ninguno; pero los del Z80 pueden ser de dos bytes de largo, seguidos también, a su vez, por uno o dos bytes de datos o por ninguno.

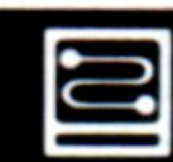
Al ser enviado un opcode al microprocesador, el programa interno de la CPU lo decodifica en códigos de operación y de longitud, y es esta última información la que le permite al microprocesador interpretar los bytes que siguen al opc. Por ejemplo, para el 6502 la secuencia de bytes hexas:

```
A9 0E 8D 01 4E 60 44 52 41 54
```

representa tres instrucciones, seguidas de cuatro bytes de códigos ASCII. Esto se podría reescribir:

```
A9 0E
8D 01 4E
60
44
52
41
54
```

que muestra cómo la primera instrucción es el opc A9, seguido siempre por un byte de datos; la siguiente instrucción es el opc 8D, también seguido por dos bytes de datos; mientras que la siguiente es el opc 60, que no necesita ningún dato pues sólo hace que la ejecución del programa se bifurque, de modo que los siguientes bytes de datos el procesador no los examina para nada. Si al microprocesador se le envía el primer byte, A9, cuando espera recibir un opc, entonces a partir de ese momento todo funciona bien. La información de cada opc asegurará que el procesador recoja el número correcto de bytes de datos para cada opc, y el siguiente byte se tratará como el siguiente opc. Sin embargo, si al procesador, esperando un opc, se le envía el segundo byte, 0E, tratará a éste como un opc, dando por resultado que la secuencia se interprete:



```
0E 8D 01
4E 60 44
52
```

que significa: opc 0E, que necesita dos bytes de datos; después opc 4E, que también pide dos bytes de datos, después opc 52, que no es un opc legal, ocasionando en el procesador el equivalente de un error de sintaxis. Esto demuestra cómo un error de interpretación inicial genera una serie de graves errores lógicos en la ejecución del programa.

Esto también demuestra claramente algunos otros puntos importantes acerca del lenguaje máquina: que no es muy amable con el usuario (al menos, al comienzo) en cuanto que resulta difícil de leer y de escribir; que es terriblemente secuencial, sin que haya nada, excepto el orden, que diferencie a una instrucción de otra; y que es literal sólo en la medida en que lo puede ser una máquina, obedeciendo instrucciones erróneas con la misma presteza que obedece las instrucciones correctas, y rechazando sólo errores de sintaxis.

Parte de esta hosquedad se puede evitar recurriendo a técnicas mnemónicas alfabéticas escritas en lugar de opcodes numéricos mientras se está componiendo el programa, y recurriendo sólo a los opcodes cuando el programa se está cargando realmente en la memoria. Estos recursos mnemotécnicos forman el *lenguaje ensamblador* (Assembler), y el proceso de traducción a opcodes numéricos se conoce como *ensamble* o *ensamblamiento*. Observe que existe una correspondencia biunívoca entre el conjunto de expresiones mnemotécnicas del lenguaje ensamblador y el de opcodes: aquél es de nivel más alto que el código de lenguaje máquina, la diferencia entre ambos es mínima.

Si reescribimos el fragmento de código de lenguaje máquina anterior en forma de lenguaje ensamblador 6502, éste adoptaría la forma siguiente:

```
0000 A9 0E       LDS #$0E
0002 8D 01 4E    STA $4E01
0005 60          RTS
```

mientras que la misma secuencia de operaciones en lenguaje ensamblador Z80 sería de esta manera:

```
0000 3E 0E       LD A,$0E
0002 32 01 4E    LD ($4E01),A
0005 C9          RET
```

La primera columna muestra las direcciones hexas en la memoria del primer byte de la línea; el opc A9 en el listado 6502, por ejemplo, está en el byte 0; el byte de página 4E en ambos listados está en el byte 4, y así sucesivamente. La siguiente columna puede contener uno, dos o tres bytes y muestra el listado en lenguaje máquina. La tercera columna empieza con una expresión mnemotécnica del lenguaje ensamblador y muestra la versión en este lenguaje del de máquina. No se moleste en este momento en tratar de descifrarlo todo: es suficiente por ahora haber visto un listado en lenguaje Assembler, y observar las diferencias entre las versiones Z80 y 6502. Mejor también que note ya que la dirección de la segunda línea aparece en forma *lo-hi* convencional en código de lenguaje máquina, pero en forma *hi-lo* "normal" en lenguaje ensamblador.

En el próximo capítulo del curso comenzaremos a examinar con detalle los opcodes y echaremos una mirada a la arquitectura del microprocesador.

Conversión a hexadecimal

Para convertir el programa Mempeek de la página 539 de modo que el contenido de los bytes se visualice en hexadecimal en lugar de en decimal, introduzca las siguientes modificaciones:

BBC Micro

Agregue:

```
3000 DEF PROCHXPRINT(NUMDEC)
3100 LOCAL X$
3200 X$ = "0123456789ABCDEF"
3300 HB = INT(NUMDEC/16):LB = NUMDEC-HB# 16
3400 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1) + " "
3500 PRINT B$;
3600 ENDPROC
```

y cambie la línea 600 por:

```
600 PROCHXPRINT(PK%)
```

Spectrum

Agregue:

```
10 LET X$ = "0123456789ABCDEF"
3000 REM*********S/R BYTE HEXA*********
3100 LET HB = INT (PK/16): LET LB = PK-HB*16
3200 LET B$ = X$(HB + 1) + X$(LB + 1) + " "
3300 PRINT B$;
3400 RETURN
```

Y cambie la línea 600 por:

```
600 GOSUB 3000
```

Commodore 64

Agregue:

```
10 LET X$ = "0123456789ABCDEF"
3000 REM*********S/R BYTE HEXA********
3100 HB = INT(PK/16):LB = PK-HB*16
3200 B$ = MID$(X$,HB + 1,1) + MID$(X$,LB + 1,1) + " "
3300 PRINT B$;
3400 RETURN
```

y cambie la línea 600 por:

```
600 GOSUB 3000
```

Estas modificaciones harán que el contenido de la memoria se visualice en hexadecimal. Aun así, a la dirección de comienzo y al número de los bytes se les debe dar entrada en decimal

Inversión para el futuro

Xerox, el mayor vendedor del mundo de aparatos de reproducción gráfica, se ha lanzado también a la conquista del campo de la automatización de oficinas

A principios de la década de los setenta, la conocida y prestigiosa empresa Xerox (tanto, que muchos dicen "xerocopias" en vez de "fotocopias") planificó un programa de investigación a gran escala para hacer realidad un sueño: el de tener disponible la información en la oficina al punto, como la electricidad o el agua corriente. Xerox creó un nuevo equipo de investigación con cheque en blanco y veló por su máxima libertad de funcionamiento estableciéndolo en Palo Alto (California), en la otra punta del país, alejado de las oficinas centrales de Xerox en Rochester (New Hampshire).

El traslado al condado de Santa Clara (California) dio sus frutos. Situado cerca del campus de la Universidad de Stanford, que contaba con un floreciente departamento de ciencia informática especializado en el estudio de la inteligencia artificial, el Centro de Investigación de Palo Alto (PARC: *Palo Alto Research Center*) atrajo a varios de los mejores cerebros de la informática. En esta comunidad cerrada, los estudiantes de talento podían pasar fácilmente de la investigación académica a la investigación comercial. El PARC se convirtió en el centro de la cultura de ordenadores, produciendo una jerga que sólo resultaba comprensible a los iniciados. Varios productos de Xerox recibieron apodos durante la misma fase de creación. La serie de micros 820, por ejemplo, recibió el nombre en clave de "Worm" (gusano), dando por hecho que habría de "comerse el Apple" (manzana).



Chris Stevens

El precedente del Lisa
Uno de los mayores éxitos del PARC fue el desarrollo del STAR, un sistema de programación que utiliza el lenguaje SMALLTALK. El STAR opera combinando programas y datos en el mismo archivo para procesamiento. La tecnología del Lisa de Apple le debe mucho a esta innovación; de hecho, la mayoría de los miembros del equipo de desarrollo del Lisa se reclutaron en el PARC

El ímpetu primordial del nuevo equipo de investigación estaba encaminado a desarrollar una red de área local (LAN: *Local Area Network*). Ahora este término ya se ha convertido en un lugar común, pero cuando Xerox construyó su primera red experimental, en Hawaii, a finales de los años sesenta, se trataba de un concepto revolucionario. Las conexiones entre una máquina de unidad principal y un terminal exigían un costoso cableado para comunicaciones de alta velocidad, y había problemas con los tendidos de cables más allá de los 20 metros. Se pudo utilizar la red pública de teléfonos conmutados, pero ésta limitaba el intercambio de datos a 9 600 baudios.

En Palo Alto el objetivo consistía en una red de velocidad razonable que enlazara entre sí ordenadores más pequeños, de modo que el usuario dispusiera de un potencial informático local para su propia máquina, así como del acceso a ordenadores más grandes, a grandes almacenamientos en disco y a otros periféricos caros como plotters e impresoras. Ésta fue la base del concepto Ethernet LAN.

En el sistema Ethernet, las conexiones se efectuaron con cable coaxial normal, que es capaz de transportar 10 millones de bits por segundo y tiene aptitud para transportar información digitalizada de sonido y de gráficos, así como datos. Además, el sistema podía alcanzar hasta 500 metros sin necesidad de amplificadores repetidores. Se podía enchufar cualquier dispositivo nuevo derivándolo de la red existente, lo que daba la máxima flexibilidad.

La red física es pasiva: los datos, de la clase que sean, se transmiten alrededor de la red y un *transceptor* actúa como el extremo frontal de cada dispositivo, determinando si el mensaje está destinado a ese dispositivo. De ser así, el transceptor decodifica el mensaje y lo presenta en una forma que pueda ser utilizada por el dispositivo, sea éste un microordenador, una impresora, un plotter, etc.

Para mediados de la década de los setenta el Ethernet ya estaba funcionando. Xerox creyó que si podía conseguir la ayuda de otros fabricantes el sistema se convertiría en un estándar para la comunicación entre ordenadores. Presentó sus diseños a la IBM, y ésta se negó a participar. Sin embargo, a la Digital Equipment Corporation le faltó tiempo para unirse al proyecto. En 1975 Xerox también se aseguró la cooperación del fabricante de chips Intel, que construyó el chip transceptor.

El Ethernet fue puesto a prueba en Suecia, en un complejo experimental de oficinas y fábricas, y al cabo de unas exitosas pruebas fue adoptado por otros fabricantes. Ahora se ha convertido en un estándar internacional de carácter oficial y fabricantes como Hewlett-Packard e ICL, de Gran Bretaña; Siemens, de Alemania, y Olivetti, de Italia, han decidido adoptarlo. Xerox alentó la aceptación del estándar vendiendo las heliografías por el precio total de mil dólares. Todos los productos Xerox se pueden conectar al Ethernet.




E R T Y U I O P
D F G H J K L
X C V B N M

mi computer

10029
9 788485 822836